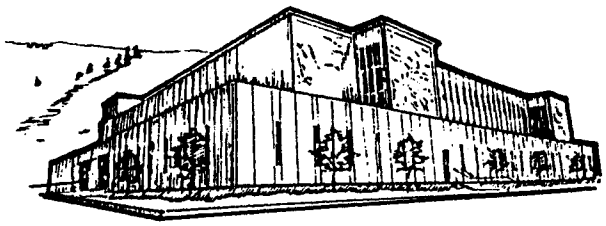
1994

Hatchery supplementation of declining populations| Fitness and role in conservation

Scott Chadde
The University of Montana



Maureen and Mike MANSFIELD LIBRARY

The University of Montana

HATCHERY SUPPLEMENTATION OF DECLINING POPULATIONS: FITNESS AND ROLE IN CONSERVATION

by

Scott Chadde

B. S. The University of Wisconsin--La Crosse, 1991

presented in partial fulfillment of the requirements

for the degree of

Master of Science

The University of Montana

1994

Approved by:

Chairperson

Dean, Graduate School

May 2, 1994

Date

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Chadde, Scott E., M.S., May 1994 Environmental Studies

Hatchery supplementation of declining populations: fitness and role in conservation (67 pp.)

Director: Fred W. Allendorf FWA

Captive-reared individuals are being used to rebuild or supplement declining populations threatened in the wild. However, little consideration has been given to the impact supplementation has on the fitness of a population. I modeled the process of supplementation, using Pacific salmon (*Oncorhynchus* spp.) as a model, to examine the effect introducing hatchery-reared individuals into a population has on fitness of that population. The model was analyzed in the context of different management options designed to minimize fitness differences between wild- and hatchery-reared salmon. All simulations showed loss of mean fitness as a result of supplementation. The amount and rate at which fitness was lost in a supplemented population were primarily a function of the intensity of selection in the captive environment. Simulations suggest that management is most effective if targeted at controlling the artificial environment, rather than management aimed at manipulating the hatchery broodstock. However, management to control the captive environment--building better artificial environments and implementing captive-rearing and genetic management techniques--are likely to be expensive options and of limited effectiveness. The results of the model suggest supplementation is effective in the short-term only, and its role in rebuilding declining populations needs reevaluating.

ACKNOWLEDGMENTS

I like to thank Fred Allendorf for teaching me population genetics and supporting this project. Rebecca Irwin for numerous discussions, free coffee, and listening. Frank Irwin for an invaluable reference.

Finally, I like to thank Jody and Sky for making it all worthwhile, and to Rio who gave me the push I needed to finish.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

INTRODUCTION

Increasingly captive-reared individuals are being used to rebuild or supplement declining populations threatened in the wild. Examples of terminated, ongoing, and proposed supplementation projects (also known as head starting, or supportive breeding) include the Houston toad (*Bufo houstonenis*), the Puerto Rican crested toad (*Peltophyryne lemur*), the green turtle (*Chelonia mydas*), the loggerhead turtle (*Caretta caretta*), the hawksbill turtle (*Eretmochelys imbricata*), the Kemp's ridley turtle (*Lepidochelys kempi*), the gharial (*Gavialis gangeticus*), the American alligator (*Alligator mississippiensis*), numerous other crocodiles (*Crocodylus* spp.) and caimans (*Caiman* spp.), chinook (*Oncorhynchus tshawytscha*) and coho salmon (*O. kisutch*), rainbow (*O. mykiss*) and bull trout (*Salvelinus confluentus*), the Bald Eagle (*Haliaeetus leucoecephalus*), the Piping Plover (*Charadrius melodus*), and the Snowy Plover (*C. alexandrinus*) (Klima and McVey 1982; Chourdhury and Chowdhury 1986; Dodd 1988; Simons et al 1988; Page et al 1989; Johnson 1990; King 1990; Clune and Dauble 1991; Dodd and Seigel 1991; USFWS 1991; NWPPC 1992; Powell and Cuthbert 1993). Moreover, the practice of supplementing populations with captive-reared individuals is likely to increase in use as a conservation strategy for declining populations given the current rate of habitat destruction and the expansion of captive propagation's role in conservation from the last measure taken to save a species, i.e., the California condor (*Gymnogyps californianus*), to a proactive strategy used in conjunction with other conservation measures to recover declining populations (Conway 1988; Griffith et al. 1989; de Boer 1992).

Unlike other captive breeding programs that maintain permanent captive populations and involve the release of individuals from permanent captive parents, most supplementation programs remove a fraction of a wild population's breeding adults, eggs, or hatchlings, raise them in captivity to increase reproductive success and/or survivorship,

and then release the captive-reared individuals into their native environment where they are expected to integrate with conspecifics. The logic of augmentation is that by increasing the survivorship in the early life stages, presumably more individuals will be recruited into the natural population as reproductive adults and contribute to the next generation; consequently, the population growth rate (*r*) increases.

However, there has been little consideration of the effect supplementation has on the fitness of the targeted population. Yet, knowing the extent supplementing populations with captive-reared individuals affects fitness of a targeted population is essential in evaluating its role in rebuilding populations as the assumption that wild- and captive-reared individuals have similar fitness in the wild--that is, similar ability to survive and reproduce--underlies the entire logic of supplementation projects. If less fit captive-reared individuals, interbreed with their wild counterparts as expected, then there will be an increase of genotypes less fit in the wild; consequently, a population's mean fitness will decrease relative to its original fitness prior to supplementation, and result in a decreased population growth rate (Crow and Kimura 1970). Thus, supplementation may actually depress the growth rate of a declining population even more, and contribute to its demise, rather than rebuild the population. The utility of supplementation as a conservation measure, then, is undermined if captive-reared individuals are less fit than their wild counterparts.

For several reasons directional genetic change in the captive environment is inevitable even during the limited time individuals will spend in captivity prior to their release into the wild and is likely to result in fitness differences between wild- and captive-reared individuals. First, environmental conditions in a artificial environment are very different from the wild--if this was not true, there would be no need for artificial propagation--and, consequently, a very different selection regime exists in the captive environment than that found in the wild (Spurway 1952; Frankel and Soulé 1981; Price 1984; Frankham et al.

1986; Allendorf and Ryman 1987; Waples 1991). Moreover, the goal of maintaining the genetic variation found in the natural population in captivity to increase the chance of success for captive-reared individuals surviving in the wild and to maintain the ability to adapt to future environments (Frankel and Soulé 1981; Foose et al. 1986; Frankham et al. 1986; Hedrick et al. 1986; Ralls and Ballou 1986; Allendorf and Ryman 1987), makes individuals responsive to selective changes in captivity. Second, captivity releases individuals from natural selection in the wild (Price 1984; Frankham et al. 1986; Allendorf and Ryman 1987; Waples 1991) as reflected in the fact captivity reverses the natural mortality pattern. For example, typically, egg-to-smolt survival rates range from 5 - 10% for wild-spawned salmonids as compared to a 60 - 80% egg-to-smolt survival rate for hatchery reared salmonids (Howell et al. 1985) (see Simons et al 1988; Page et al 1989; King 1990; Waples 1991; Powell and Cuthbert 1993 for other specific examples). Thus, genotypes survive in captivity that normally would not have survived in the wild. Third, traits selected for in artificial environments are usually detrimental in the wild (Spurway 1952, 1955; Frankel and Soulé 1981; Price 1984; Frankham et al. 1986; Kohane and Parsons 1988). For instance, risk of predation influences a host of behavior decisions such as foraging, mating, and social behaviors (see Lima and Dill 1990 for a review), and individuals raised in a predator-free environment may exhibit maladaptive behaviors in the wild that reduce their fitness (Price 1984; Lyles and May 1987). For example, compared to wild fish hatchery, coho (*Oncorhynchus kisutch*) and hatchery x wild rainbow (*O. mykiss*) hybrids have been found to have increased aggressiveness in foraging behavior, a trait shown to have a genetic basis, even in the presence of predators (Swain and Riddell 1990; Johnsson and Abrahams 1991).

Furthermore, empirical studies support the idea that even limited exposure to captivity reduces the fitness of captive-reared individuals relative to wild conspecifics. Studies have shown reduced disease resistance in coho after seven months of hatchery experience

(Salonius and Iwama 1993), decreased survivorship in Atlantic salmon (*Salmo salar*) after only a year of hatchery culture (Jonsson et al. 1991), and decreased survivorship in the offspring of naturally spawning hatchery x hatchery steelhead (*Oncorhynchus mykiss*) and hatchery x wild matings compared to the offspring of wild x wild matings after only two generations of hatchery culture (Reisenbichler and McIntyre 1977).

In this paper, I present a model that simulates the process of supplementation, using Pacific salmon (*Oncorhynchus* spp.) as a model, to test the effect of introducing captive-reared individuals into a population has on fitness of that population. For several reasons, Pacific salmon are an ideal group of species to evaluate the effect of supplementation on the fitness of a targeted population. First, Pacific salmon populations are probably the most extensively supplemented populations of all species. For example, the current Columbia River Basin Fish and Wildlife Program calls for the doubling of salmon production from approximately 2.5 million returning adults to 5 million returning adults and more than 50% of this projected increase is to come from supplementation programs (NWPPC 1992; RASP 1992). Second, the process of hatchery-rearing salmon mirrors the process of supplementation in general. Typically, salmon spend only a portion of their life cycle in the hatchery, usually a year, before they are released into the wild and the hatchery broodstock is collected from adults returning from the ocean. Third, salmon are a species with high individual fecundity which is typical of the majority of species that are the most likely candidates for supplementation. Finally, a number of untested management recommendations have been suggested for salmon that if shown to be effective in minimizing fitness differences between wild- and hatchery-reared fish would be applicable to other supplementation programs. These include (1) select only individuals of wild origin to spawn in the hatchery rather than hatchery-reared individuals (2) limit the proportion of the population brought into the hatchery to spawn, (3) build better hatcheries so that the hatchery environment mimics as much as possible the natural environment's selection

regime, and (4) implement better hatchery-rearing practices and genetic management of hatchery broodstock (Reseinbichler and McIntyre 1977, 1986; Helle 1981; Nickelson et al 1986; Clune and Dauble 1991; Cuenco et al. 1993). The model was analyzed in the context of these different management options designed to minimize fitness differences between wild- and hatchery-reared salmon. I examine the sensitivity of the model to different parameter values, and analyze the impact these parameters have on loss of fitness in an supplemented population.

MODEL

Assumptions

Three fundamental genetic assumptions were made in constructing the model:

(1) Selection in the hatchery is inevitable, and will increase the frequency of some genotypes that are less fit in the wild.

(2) Alleles that increase fitness in the hatchery, but are harmful in the wild, occur at low frequency in wild populations. In addition, the effects of these alleles are likely to be recessive in the wild, otherwise they would be eliminated by natural selection. Such alleles are maintained in the wild at low frequencies by a balance between natural selection and mutation or a balance between natural selection and migration.

(3) Only alleles that are additive or dominant in their effect in the hatchery will respond quickly to selection in the artificial environment. This follows directly from the previous assumption that these alleles are rare in the wild. Rare recessive alleles respond extremely slowly to even strong selection; therefore, it is assumed that the alleles of interest are not recessive in their effects on fitness under artificial conditions.

Finally, it is assumed that supplementation is successful. That is, hatchery-reared individuals released into the wild successfully return as adults to spawn and potentially contribute to future generations in the wild.

The model

These assumptions were incorporated into a discrete-generation, Monte Carlo simulation model that used a stylization of the life cycle of salmonids and the supplementation process as its structuring algorithm (Fig. 1). The model kept track of an individual's place of birth (wild or hatchery) and genotype at, depending on the simulation, one, three, or five unlinked loci. Each locus has two unique alleles and no mutation occurs. At each locus one allele, x_1, is favored by selection in the wild, and the alternative allele at each locus, x_2, is favored by selection in the hatchery. The initial allele frequency of x_1 at each locus is 0.95.

Initially the population consisted of N ($N = 2000$) random mating wild returning adults. Each generation, a proportion of the returning adults, $(1 - \beta)N$, were chosen to spawn in the hatchery, and the rest of the returning adults, βN, spawned in the wild. Returning adults were selected for either spawning environment by comparing a uniformly distributed [0, 1] random number with β for all individuals. Two types of hatchery broodstock were simulated: (1) mixed origin, where each generation both wild- and hatchery-born returning adults spawned in the hatchery; and (2) wild origin, where each generation only of wild-born adults spawned in the hatchery.

Random mating occurred within each environment (wild and hatchery), and zygotes were created by simulating Mendelian inheritance. Viability selection occurred within the different environments and whether or not a zygote survived to adulthood was determined by comparing a uniformly distributed [0, 1] random number with the fitness of the zygote. A zygote's fitness depended on which environment it was in. In the wild, the relative

fitness (w_{11}, w_{12}, w_{22}) of the three genotypes (x_1x_1, x_1x_2, x_2x_2) at each locus was 1, 1, and 1 - s. In the hatchery, the relative fitness (w_{11}, w_{12}, w_{22}) of the three genotypes was 1 - s, 1 - sz, and 1 (Table 1). Thus, s is the selection coefficient against the "hatchery" allele in the wild as well as the selection coefficient against the "wild" allele in the hatchery. z determines the dominance relationship of the heterozygote in the hatchery. In multiple-loci simulations, multiplying the fitness of each genotype at each locus together determined the zygote's fitness.

In the wild, βN spawners produced βN returning adults for the next generation, and in the hatchery, $(1 - \beta)N$ spawners produced $(1 - \beta)N$ returning adults for the next generation. That is the model did not account for the expected increase in natural production due to supplementation. Further, it was assumed that no selection occurred from surviving zygote to returning adult. Once all of the next generation returning adults were produced, both hatchery-born and wild-born returning adults were grouped into one wild returning adult population, and their fitness calculated. However, unlike the fitness calculation for viability selection, the fitness of adult genotypes did not depend on natal environment. That is, the relative fitness (w_{11}, w_{12}, w_{22}) of the three genotypes at each locus (x_1x_1, x_1x_2, x_2x_2) was 1, 1, and 1 - s whether an individual was born in the wild or in the hatchery (Table 1). In multiple-loci simulations, multiplying the fitness of each genotype at each locus together determined the zygote's fitness. The mean fitness (ϖ) of the supplemented population was then calculated as $\left(\sum_{i=1}^{N} f_i\right)/N$ where f_i is the fitness of the ith individual in the wild and N is the total number of returning adults. Thus, mean fitness is a summary statistic of the supplemented population's ability to produce and survive in the wild relative to a non-supplemented population, whose mean fitness is 1.

Simulations

Three different management strategies were simulated: (1) each generation, 50% of the returning adult population spawned in the hatchery ($\beta = 0.50$) and both hatchery- and

wild-born adults comprised the hatchery broodstock; (2) each generation, 25% of the returning adult population spawned in the hatchery ($\beta = 0.75$) and both hatchery- and wild-born adults comprised the hatchery broodstock; and (3) each generation, 25% of the returning adult population spawned in the hatchery ($\beta = 0.75$) and the hatchery broodstock consisted only of wild-born adults. The case where 50% of the adult population spawned in the hatchery and the hatchery broodstock consisted only of wild-born adults was not examined since the dynamics of the model treated the different groups of adults as two separate and distinct populations, rather than as one population.

For each of the different management strategies, the following parameters were varied: (1) the number of loci under selection, (2) the strength of selection ($1 - s$) in the wild against genotypes favored in the hatchery, and (3) the fitness of the heterozygote in the hatchery ($1 - sz$). These parameters were examined in a hierarchical fashion. Simulations were run with one, three or five unlinked loci under selection for each individual in the population. For each case of the number of loci under selection, three different levels of the intensity of selection were examined: weak ($1 - s = 0.75$), intermediate ($1 - s = 0.50$), and strong ($1 - s = 0.25$) selection. Finally, for each case of the intensity of selection, simulations were run for $z = 0$ in which case the relative fitness of the heterozygote in the hatchery was 1 and the x_2 allele was dominant in the hatchery, and for $z = 0.5$ in which case and the relative fitness of the heterozygote in the hatchery was additive.

Also, for each management strategy, simulations were run to see how long it would take a supplemented population, once supplementation ceased, to regain, 99% of its original fitness. The value 99% of its original fitness was arbitrarily set. Supplementation was stopped after $0.25t_{eq}$, $0.50t_{eq}$, $0.75t_{eq}$, and t_{eq} generations had elapsed, where t_{eq} is the number of generations it takes a supplemented population to reach its new equilibrium fitness.

MODEL RESULTS

All simulations showed a decline in mean fitness of a supplemented population relative to a non-supplemented population because of the increase in frequency of alleles favored under artificial propagation (Fig. 2; Fig. 3). In general, reductions in mean fitness of a supplemented population increased as the percentage of the population used as hatchery parents (1 - β), the number of loci under selection, and the intensity of selection (1 - *s*) increased (Fig. 2; Fig. 3).

Proportion of the population used for hatchery broodstock

The effect the proportion of the population used as hatchery broodstock (1 - β) had on mean fitness was mixed. For a given number of loci under selection and a given intensity of selection, mean fitness of a supplemented population declined more when 50% of the population spawned in the hatchery, than when 25% of the population spawned in the hatchery (Fig. 2; Fig. 3). However, for different selection coefficients, different values of β produced similar declines in mean fitness (Fig. 3).

Number of loci under selection

As expected, for a given intensity of selection reduction in mean fitness increased as the number of loci under selection increased (Fig. 2; Fig. 3). This is a direct consequence of assuming multiplicative fitness in the algorithm. However, while increasing the intensity of selection magnified the differences between the number of loci under selection, this effect was dependent on the proportion of the population used as hatchery broodstock. Simulations where 50% of the population spawned in the hatchery exacerbated differences in the number of loci under selection with increasing intensity of selection; while simulations where 25% of the population mated in the hatchery did not produce such great differences (Fig. 2; Fig. 3).

Type of hatchery parent population

The natal origin of hatchery broodstock--either all wild-born returning adults or a mix of wild- and hatchery-reared adults--made relatively little difference in the reduction of mean equilibrium fitness (Fig. 4). However, natal origin of hatchery broodstock affected the rate fitness was lost, and thus in preventing selective changes in the hatchery. This effect increased as the number of loci under selection and the intensity of selection increased (Fig. 4).

Effect of the allele in the hatchery environment

Loss of mean fitness was relatively insensitive to the effect of the hatchery-favored allele, either additive or dominant, in the hatchery environment. Simulations where 50% of the population mated in the hatchery, dominant selection ($z = 0$) occurred in the hatchery, and 5 loci under selection reduced mean equilibrium fitness by 25%, 42%, and 46% for weak, intermediate, and strong selection, respectively. For similar simulations, additive selection ($z = 0.5$) in the hatchery, reduced mean equilibrium fitness by 29%, 51%, and 57% for weak, intermediate, and strong selection, respectively. Like the type of hatchery parent population used, reductions in mean fitness due to differences between dominant and additive fitness in the hatchery increased as intensity of selection against the alleles favored in the hatchery increased in the wild.

Rate of loss of mean fitness and intensity of selection

Mean fitness was lost at a much faster rate as the intensity of selection increased (Fig. 5). This result was independent of the number of loci and, to a large degree, the proportion of the population used as hatchery parents. Simulations involving weak selection lost mean fitness gradually over time (Fig. 4). Mean fitness was lost relatively rapidly for simulations involving intermediate and strong selection, with the greatest declines in mean fitness occurring with strong selection (Fig. 4). Appreciable amounts of

mean fitness began to be lost from a supplemented population by the seventh, third, and second generation for weak, intermediate, and strong selection respectively.

A similar pattern held for the time it took a supplemented population to reach a new equilibrium. Simulations where 50% of the population bred in the hatchery and 5 loci where under selection reached a new equilibrium after 56, 24, and 14 generations for weak, intermediate, and strong selection, respectively; while simulations involving 25% of the population mating in the hatchery (wild only) and 5 loci under selection reached a new equilibrium after 54, 31, and 15 generations for weak, intermediate, and strong selection, respectively. This result was independent of the number of loci, and the proportion of the population used as hatchery parents. This result suggests that the qualitative dynamics of supplementation is determined by the intensity of selection. The number of loci under selection, the proportion of population used as hatchery parents, and the intensity of selection determine the absolute loss in mean fitness.

Return time

The number of generations (t) it took for a supplemented population to return to 99% of its original fitness, after supplementation ceased, varied as a function of selection intensity (Fig. 6). The dynamics of restoration after supplementation was independent of the proportion of the population spawned in the hatchery, the number of loci under selection, and the time supplementation was ended. While quantitatively, populations returned to their original fitness quicker when supplementation was stopped earlier and involved fewer individuals spawning in the hatchery, qualitatively the dynamics of return were the same whether supplementation ceased after $0.25t_{eq}$, $0.50t_{eq}$, $0.75t_{eq}$, or t_{eq} and whether 25% or 50% of the population spawned in the hatchery. At weak selection, it takes a relatively long time for a supplemented population to return to 99% of its original fitness after supplementation stops (Fig. 6). At strong selection, a supplemented

population returns relatively quickly to its original fitness, and at intermediate selection, the return is intermediate (Fig. 6).

DISCUSSION

Supplementation model

The simulations strongly suggest that if supplementation is successful and hatchery-reared individuals interbreed with wild conspecifics, then potentially supplementing populations with hatchery-reared individuals can have a considerable negative impact on the mean fitness of a population. Supplementation, then, may actually decrease a population's growth rate, rather than increase it and exacerbate the problems of an already declining population.

The results of the model strongly suggest that the effect supplementation will have on mean fitness of the targeted population depends on the intensity of selection in the captive environment. This result is emphasized by the fact that the greater the fitness difference between wild- and hatchery-reared individuals, the greater the loss of mean fitness in the simulations, that simulations produced similar declines in mean fitness with different proportions of the population used as hatchery broodstock, and that strong selection in the hatchery increased the rate at which mean fitness was lost in a supplemented population. Moreover, this result is general and applies to other supplementation projects where eggs (e.g., crocodiles) or hatchlings (e.g., sea turtles) are reared in captivity rather than breeding adults since the assumptions of the way selection operates in captivity is independent of the life-stage involved.

The utility of supplementation as a conservation measure, then, depends on how well management can control against selective changes in captivity. The importance of management to minimize selective changes in the captive environment is further highlighted by the fact that theory and empirical evidence indicate that rapid selective

changes are possible within limited exposure to captivity, suggesting strong selection is involved in artificial environments.

On theoretical grounds, it can be effectively argued that rearing animals in artificial environments is a form of environmental stress (Kohane and Parsons 1988) and rapid genetic changes are most likely under stress (Belyaev 1979; Parsons 1986; Kohane and Parsons 1988), implying that potentially strong selection can occur within artificial environments. Second, as mentioned, empirical studies have documented rapid changes in hatchery-reared fish resulting in deficiencies in traits with a genetic basis that affect performance (Reisenbichler and McIntyre 1977; Jonsson et al. 1991; Salonius and Iwama 1993), suggesting strong selection is involved in the hatchery. Other studies have documented changes as well. Verspoor (1988) found that after one generation of hatchery culture Atlantic salmon had 28% less heterozygosity and 12% less allelic diversity than their wild counterparts. While reduction in genetic variation cannot be said to directly effect fitness, since it is assumed to be selectively neutral, this study is indicative of the massive genetic changes possible within limited exposure to the hatchery. Moreover, rapid adaptation to captivity has been shown in *Drosophila* (Frankham and Loebel in press), and Myers and Sabath (1980) have shown the success of the release of insects for biological control depends on the amount of time spent in captivity. Thus, available evidence indicates that fitness differences are likely between wild- and hatchery-reared individuals, suggesting supplementation is more likely to harm a population, than to benefit it.

Management options designed to minimize selective changes in captivity fall into two broad categories: (1) management aimed at manipulating the number and type of individuals brought into captivity for rearing, and (2) management aimed at controlling the artificial environment through the design and construction of captive-rearing facilities and implementation of captive-rearing and genetic management techniques.

Management of the hatchery broodstock

Several authors have suggested that limiting the number of individuals of wild origin brought into the hatchery should minimize differences between wild- and hatchery-reared individuals (Reisenbichler and McIntyre 1977, 1986; Nickelson et al 1986; Clune and Dauble 1991; Cuenco et al. 1993; Marsden et al. 1993). These ideas were partially supported by the simulations. The model is sensitive to the proportion of the population brought into the hatchery, suggesting the absolute number of individuals reared in captivity prevents selective change, and in general, the less individuals used as hatchery broodstock, the less impact will there be on the targeted population's mean fitness. However, this result carries with it two caveats. First, it holds only for weak, and intermediate selection. The effects of strong selection are not buffered by the number of individuals brought into the hatchery. Second, in the model the proportion of individuals brought into the hatchery is the same as the proportion of hatchery-reared individuals in the supplemented population. If supplementation is successful, limiting the number of individuals brought into the hatchery will not have the same effect predicted in the model, since there will be an expected increase in the number of hatchery-reared individuals in the wild, and hence, a greater proportion of individuals in the wild will be hatchery-reared, and the model would predict a greater loss of fitness.

Likewise, similar amounts of mean equilibrium fitness were lost whether the hatchery broodstock consisted of only wild-born individuals, or a mix of wild- and hatchery-reared individuals, suggesting the effectiveness of this management option is of limited value. However, returning hatchery-reared individuals to the hatchery affected the rate at which fitness is lost, suggesting that in the short-term bringing only individuals of wild origin into captivity might be effective in minimizing the loss of mean fitness in the targeted population. Yet, this effect was dependent on the intensity of selection, again highlighting the need to control the intensity of selection that occurs in the hatchery.

Thus, the model predicts that management designed to manipulate the number and kind of individuals reared in the hatchery is only effective to the extent the intensity of selection is controlled for in the captive environment, and will not in itself prevent the loss of mean fitness in a supplemented population. In other words, these management options are only effective within the context of other management options and considerable loss of mean fitness could occur despite their implementation if other measures are not taken.

Management of the captive environment

Ultimately, then, the utility of supplementation as a conservation depends on the effectiveness of management aimed at preventing selective changes in the captive environment by controlling the captive environment itself, rather than management aimed at manipulating the composition of the hatchery broodstock. Management options aimed at controlling selection in the captive environment include (1) building improved captive-rearing facilities, and (2) implementing captive-rearing and genetic management techniques.

Minimizing selection in captivity through the design and construction of a captive-rearing facility involves considerable cost. For example, construction cost for a proposed hatchery to supplement populations of salmon and rainbow trout on the Yakima River in Washington is estimated at over $31 million (Clune and Dauble 1991). And given that salmon are locally adapted, this hatchery is limited to supplementing salmon stocks on the Yakima river, rather than the entire Columbia River basin. Likewise, different species have different spatial and temporal needs, thus eliminating to a large degree generic captive-rearing facilities for other species as well.

Furthermore, it is impossible to replicate the natural environment in a captive environment; thus, ensuring selection in captivity will not mirror natural selection in the wild, and there will be an altered selective regime in the captive environment. Moreover, a species reproductive ability affects the rate genetic change occurs in artificial environments

and rapid adaptation to artificial environments is more likely in species with high individual fecundity, such as many amphibians, fish, and reptiles--the most likely groups of species targeted for supplementation--than in species with low individual fecundity, such as birds and mammals. Thus, building better captive-rearing facilities is no insurance that strong selective changes will not occur in the captive environment, particularly with species with high individual fecundity like salmon.

A number of captive-rearing practices potentially affect the performance of captive-reared individuals in the wild--when and how adults and eggs are collected (time of return spawn in salmonids), type of food used and how individuals are fed (foraging and predator behavior), temperature (sex determination in turtles), water flow (survival and growth rates in salmonids), stocking density (disease control and behavior), location and timing of release (competition with wild individuals and increased predation), to name a few. However, management to prevent these changes from occurring in captivity is intensive, costly, and difficult to implement and, like building better captive-rearing facilities, only partially successful at preventing selective changes. Management of high fecund species is further hampered since the sheer number of individuals reared in captivity prevents the application of some techniques such as predator-avoidance training as in captive-reared birds (e.g., Ellis et al 1978) is impractical.

One potentially useful genetic management technique that has been suggested to minimize adaptation to artificial environments and maintaining reproductive fitness is equalizing family size, i.e., equalizing reproductive effort of individuals (Frankel and Soulé 1981; Foose et al. 1986; Ralls and Ballou 1986; Borlase 1993; Allendorf 1993). While this technique has been suggested for permanent captive populations it should be employed in supplementation projects as well since it would avoid the problem of favoring the greatest reproductive effort in captivity to the most "domestic-like" individuals, and eliminates fitness differences between captive-reared individuals. Yet, this suggestion

comes with two caveats. First, Allendorf (1993) found that in general the effectiveness of this technique holds only when weak selection is involved. Second, equalizing family size only slows down the rate of adaptation to captivity; it does not prevent it from happening altogether and is likely to an expensive technique to employ.

Thus, while management to control the captive-environment is likely to minimize the fitness differences between wild- and captive-reared individuals, fitness differences between the two can still be expected, particularly with species with high individual fecundity. Further, the implementation of management to control the artificial environment is costly, and difficult to employ, lessening the effectiveness of these management options to minimize fitness differences. Under the best of circumstances, then, it is likely supplementation will impose some cost to a natural population.

Supplementation and loss of genetic variation

Even if management is successful at minimizing fitness differences between wild- and hatchery-reared individuals, the model predicts that once supplementation ceases a population's recovery of its original fitness would take considerably longer than in the case where management unsuccessfully minimized differences between wild- and hatchery-reared individuals. That is, within the assumptions of the model, it takes a relatively long time for natural selection to eliminate individuals slightly less fit, than it does for selection to purge a population of individuals with extreme fitness differences. Thus, paradoxically the penalty for a successfully controlling the intensity of selection in captivity, and therefore, increasing the success of a supplementation program is that it potentially has long-term negative effects on the natural population.

One possible long term consequence of a successful supplementation program is the loss of genetic variation within a population, and hence the ability of a population to adapt to changing environments. Minimizing fitness differences between wild- and captive-reared individuals suggests that supplementation is most likely to be successful and

captive-reared individuals will survive to contribute to future generations. However, captive-reared individuals can still be expected to be less fit than their wild counterparts, and thus differ in there expected contribution to the future generations. Consequently, the effective population size will be smaller than the census size (Hartl and Clark 1989), and genetic variation will be lost as a direct result of supplementation. Moreover, the greater the variance in fecundity between hatchery and wild individuals, the smaller the effective population size (Hartl and Clark 1989). This result holds even when supplementation increases the total production of offspring in a population (Ryman and Laikre 1991). Thus, genetic variation can be expected to be lost during supplementation (Ryman and Laikre 1991) and, if successful, long after supplementation ceases. Supplemented populations, then, are likely to diverge genetically from their original state.

Supplementation's role in conservation

The results of the model indicate that supplementing declining populations with hatchery-reared individuals is most effective in the short-term. First, given the potential for strong selection in the captive environment, Supplementation is most likely to impose an additional cost to an already declining population in terms of loss of mean fitness, and loss of genetic variation. Second, management to control selective changes in the artificial environment are expensive, and difficult to implement, and will not prevent selective changes from occurring within the captive environment. Third, the model suggests that loss of fitness in a supplemented population is cumulative and the longer supplementation continues, i.e., the more captive-reared individuals enter the population, the greater the amount of fitness is lost in the population. Thus, supplementation effectiveness as a conservation tool decreases the more times a population is supplemented with captive-reared individuals. One way to minimize the negative effects of supplementation, then, is to limit the number of times a population is supplemented. Even if this analysis is wrong and selection is successfully controlled in the captive environment as some authors have

suggested it can be (e.g., Miller et al. 1988), from a conservation perspective it is better to assume that strong selection will occur in the captive environment and be wrong, then to assume weak selection will occur and be wrong.

Given that supplementation is most effective in the short-term, its role in rebuilding declining populations needs to be reevaluated and its implementation approached with the utmost circumspection. First, it needs to be recognized that supplementation can decrease the mean fitness of a population, and further depress an already declining population. Second, from a demographic standpoint it is not entirely clear that increasing survivorship of early life-stages--the demographic life-stage targeted in supplementation--will result in increased growth rate. For example, Crouse et al. (1987) found that for loggerhead turtles increasing survivorship of eggs is not as effective as increasing survivorship of juveniles loggerhead turtles in increasing the population's growth rate. This is not to say, increasing survivorship of eggs is not important, but the intended result of increasing a population's growth rate by increasing egg survivorship may not occur. Finally, the extreme expense of building an artificial environment for use in supplementation almost precludes the use of supplementation as a conservation tool since once the structure is in existence, it seems unlikely, given the effort and expense involved, its use will be restricted to only one or two times. For instance, in the Pacific Northwest numerous stocks of salmonids have declined or have gone extinct despite and, in part because of over a century of hatchery production (Nehlsen 1991; Frissell 1993), and there exists the real danger that supplementation may become a justification for the continued use of hatchery production in the region. This point is further illustrated with the fact the original proponents of head starting Kemp's ridley sea turtle now oppose the project, but the project has taken a life of its own despite the opposition (Taubes 1992).

Supplementation needs to be viewed as a captive population technique useful in limited circumstances, rather than as a replacement for habitat loss. Otherwise,

conservation efforts are mistakenly focused on the symptoms of population decline, rather than the causes of population decline. Ultimately, the success of supplementation is dependent on there being suitable habitat for species to survive and reproduce in. First, and foremost, every effort should be directed at increasing a population's growth rate through the protection and maintenance of existing habitat and the restoration of degraded habitat, and secondarily through supplementation.

Table 1. Relative fitness of the three possible genotypes at each locus in different environments. Relative fitness in the wild is for both zygotes and adults. Relative fitness in the hatchery is for zygotes only. x_1 is the allele favored by selection in the wild. x_2 is the allele favored by selection in the hatchery. s is the selection coefficient against the hatchery-favored allele in the wild as well as the selection coefficient against the wild-favored allele in the hatchery. z determines the dominance relationship of the heterozygote in the hatchery.

	Relative Fitness	
Genotype	**Wild**	**Hatchery**
x_1x_1	1	$1 - s$
x_1x_2	1	$1 - sz$
x_2x_2	$1 - s$	1

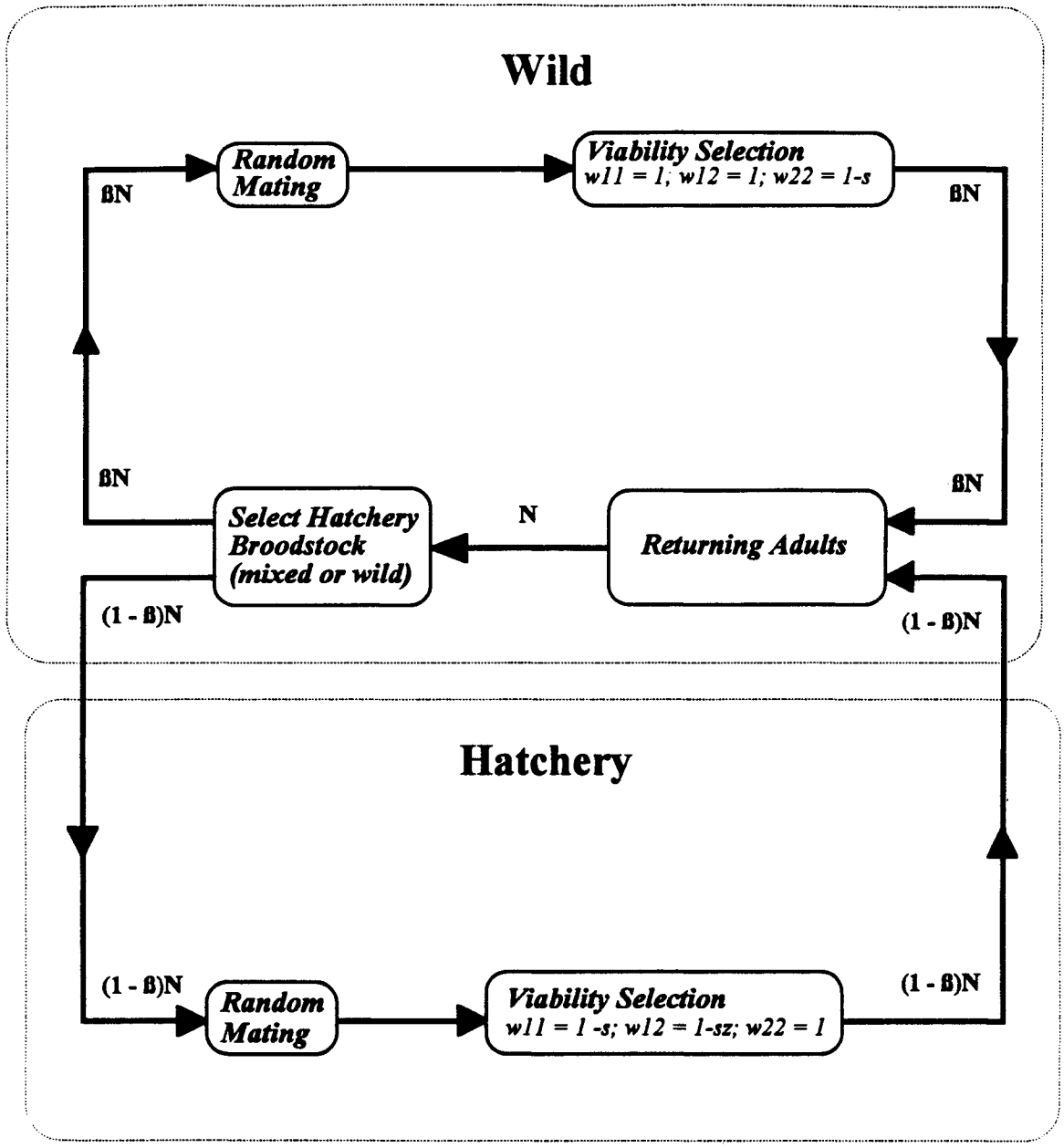


Fig. 1. Stylized life cycle of salmonids and supplementation process used to structure the model. Dashed boxes represent different spawning environments. Shaded boxes represent life history events. Solid line with arrows represent flow of individuals. N is the total number of individuals in the population. β is the proportion of the population that spawns in the wild. The hatchery broodstock for the next generation consists of either both wild- and hatchery-born individuals (*mixed*) or wild-born adults only (*wild*). Mean fitness of the population is calculated for returning adults using the fitness set: $w_{11} = 1$; $w_{12} = 1$; $w_{22} = 1 - s$.

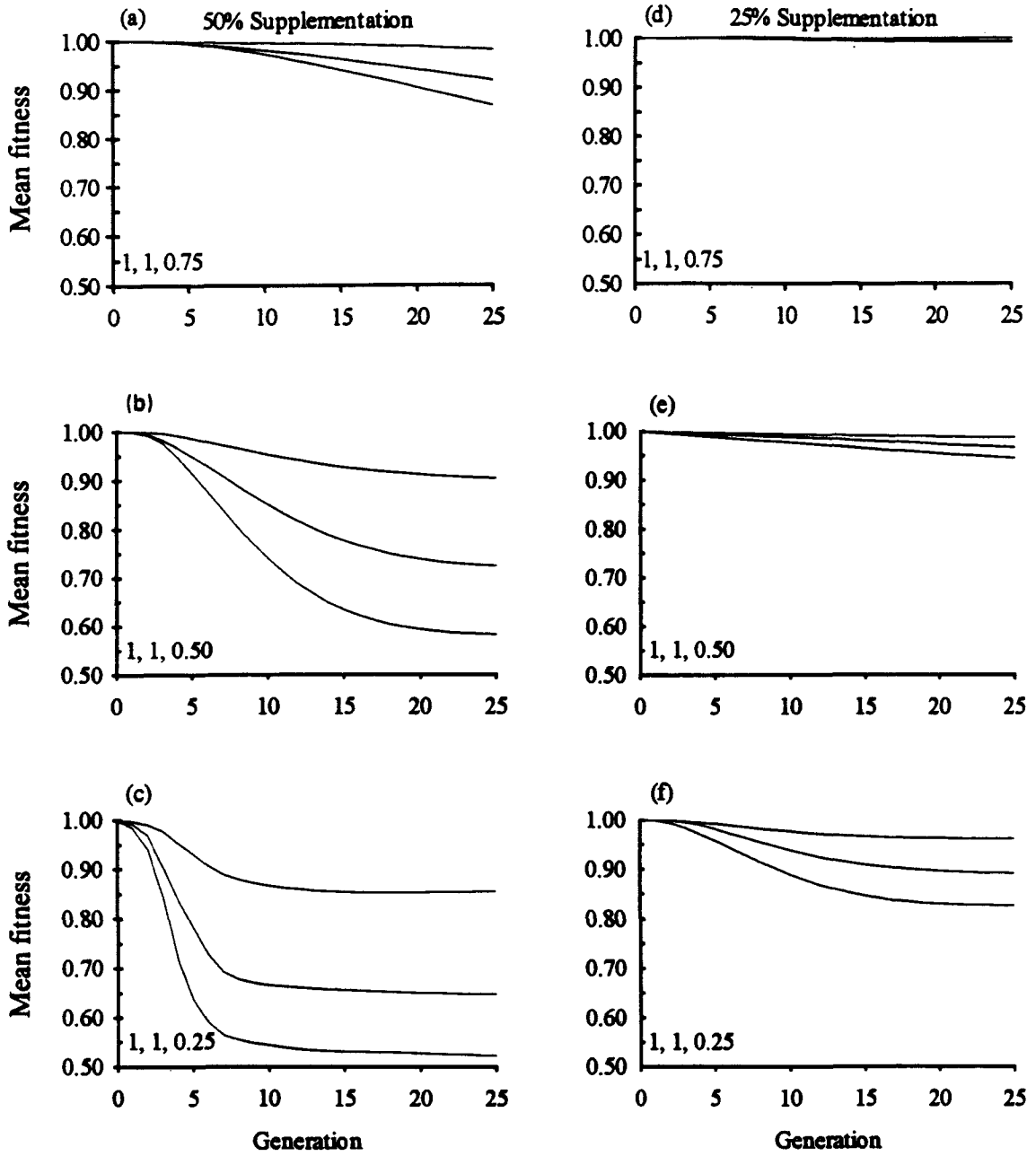


Fig. 2. Expected reduction in mean fitness of a supplemented population for two different management options. Figs. a-c, each generation, 50% of the returning adults spawn in the hatchery ($\beta = 0.50$) and the hatchery broodstock consists of both wild- and hatchery-born individuals. Figs. d-f, each generation, 25% of the population spawns in the hatchery ($\beta = 0.75$) and only wild-born individuals spawn in the hatchery. 1, 1, x are the relative fitness of the genotypes at each locus in the wild. Selection occurs at one (upper line), three (middle line), and five loci (lower line). The fitness of the heterozygote(s) in the hatchery is 1.

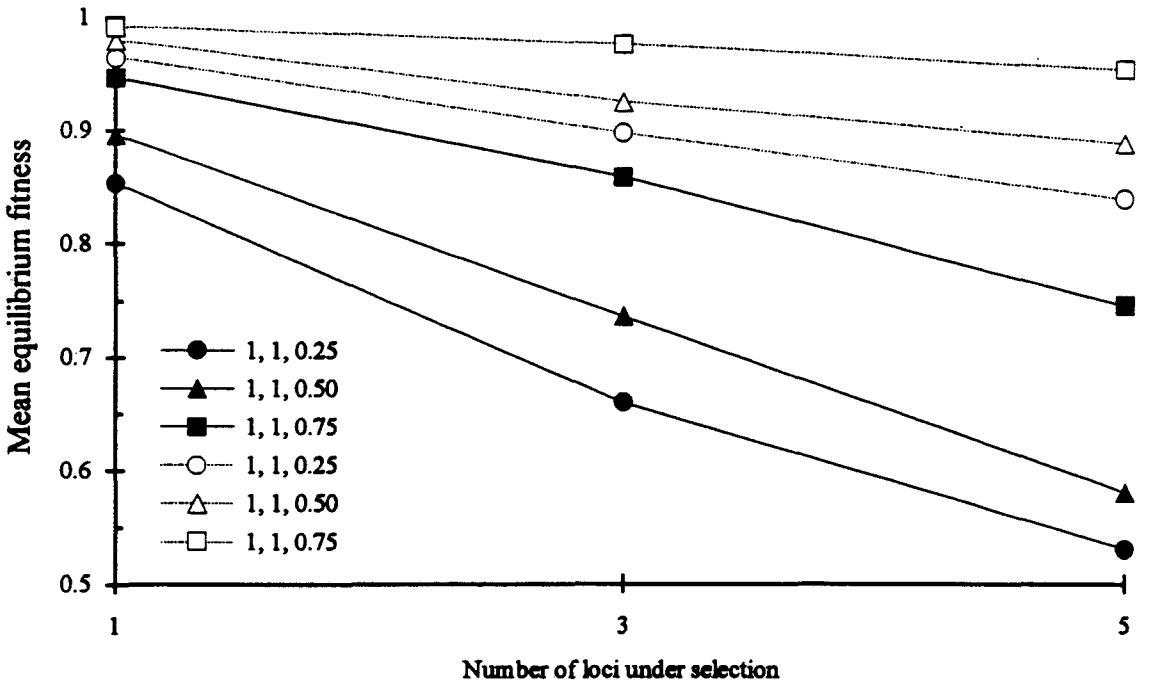


Fig. 3. Comparison of mean equilibrium fitness of a supplemented population for two different management strategies: (1) each generation, 25% of the population spawns in the hatchery ($\beta = 0.75$; ——) and hatchery broodstock consists only of wild-born returning adults; and (2) each generation, 50% of the population spawns in the hatchery ($\beta = 0.50$; ——) and both hatchery- and wild-born returning adults spawn in the hatchery. 1, 1, x are the relative fitness of the genotypes at each locus. The fitness of the heterozygote(s) in the hatchery is 1. Selection occurs at 1, 3, and 5 loci.

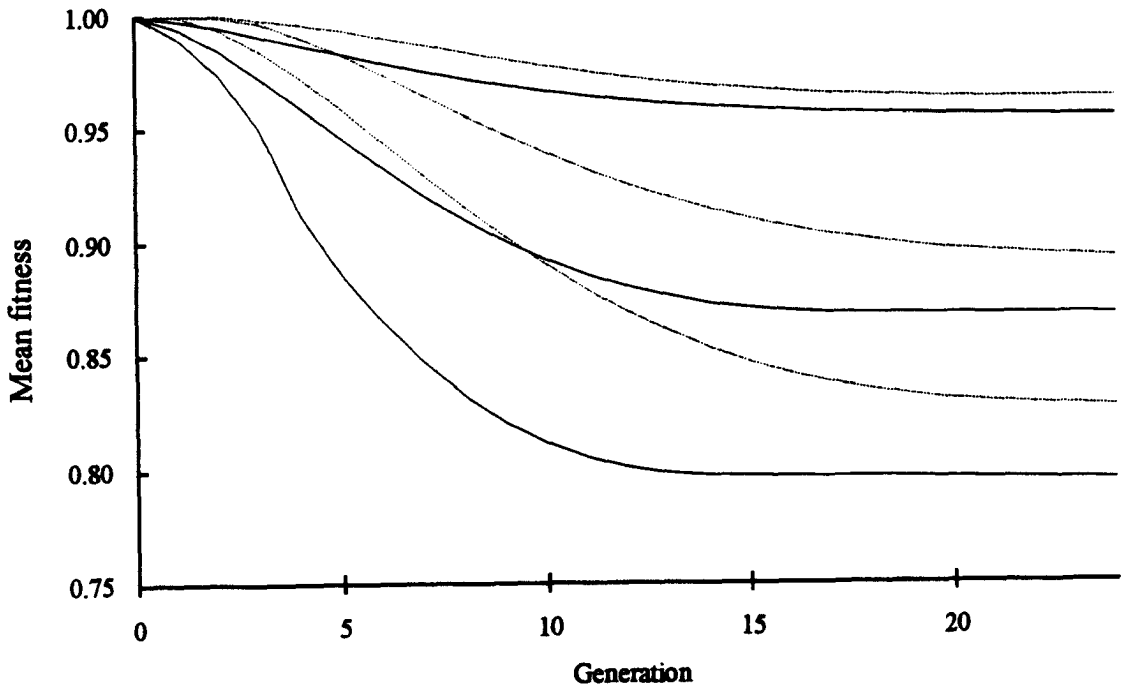


Fig. 4. Comparison of the effect type of hatchery broodstock (either wild-born only ·····; or hatchery- and wild-born ———) has on the reduction of mean fitness of an supplemented population. Each generation, 25% of the population spawns in the hatchery ($\beta = 0.75$). The relative fitness at each locus of the genotypes in the wild are 1, 1, 0.25. Selection occurs at one (upper two lines), three (middle two lines), and five loci (lower two lines).

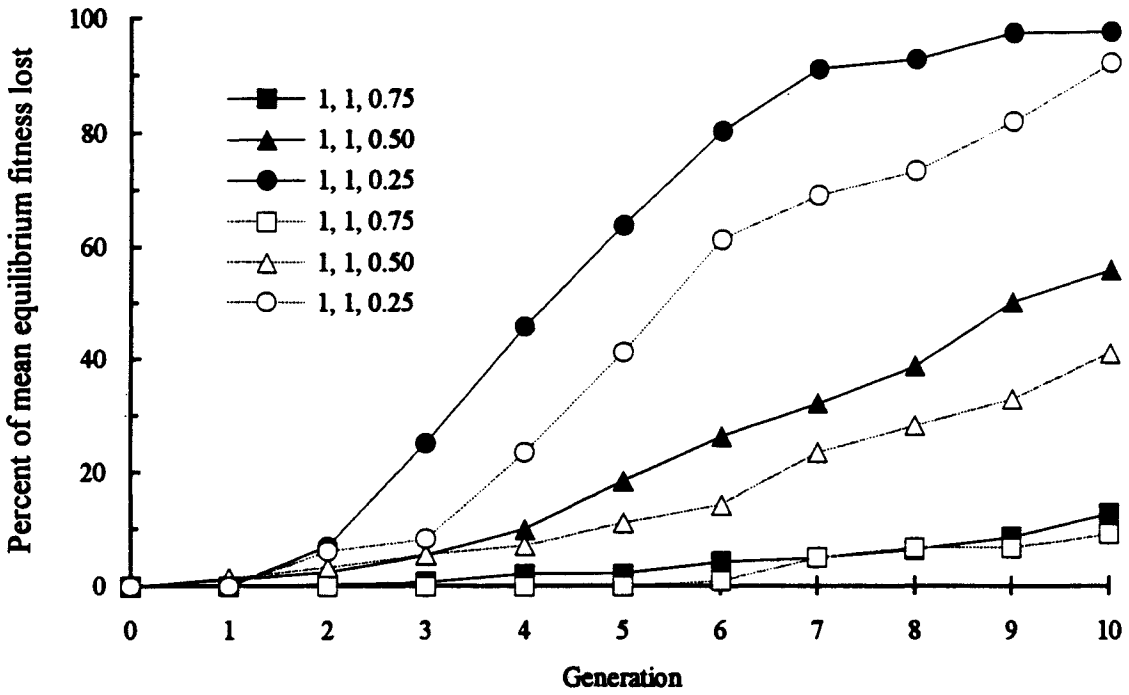


Fig. 5. Rate at which fitness is lost in an supplemented population for two different management options: (1) each generation, 25% of the returning adults spawn in the hatchery ($\beta = 0.75$; ——) and only wild-born returning adults spawn in the hatchery; and (2) each generation, 50% of the returning adults spawn in the hatchery ($\beta = 0.50$; ——)and both hatchery- and wild-born returning adults spawn in the hatchery. 1, 1, x are the relative fitness of the genotypes at each locus in the wild. Selection occurs at three loci.

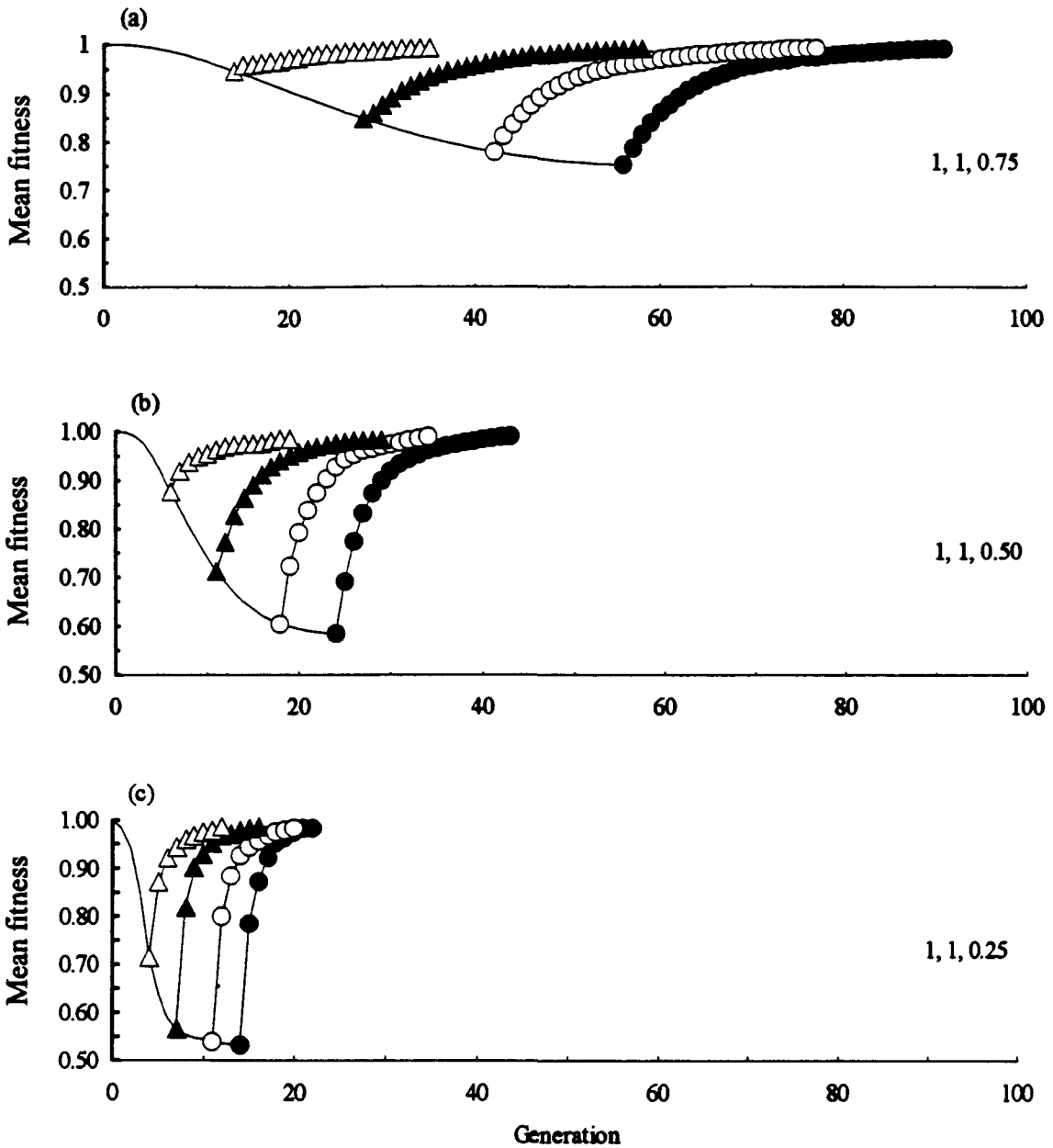


Fig. 6. Comparison of the time it takes a supplemented population to return to 99% of its original fitness after supplementation ceases. Supplementation is stopped after $0.25t_{eq}$, $0.50t_{eq}$, $0.75t_{eq}$, and t_{eq} where t_{eq} is the time it takes a supplemented population to reach its new equilibrium value. Each generation, 50% of the returning adults spawn in the hatchery ($\beta = 0.50$) and the hatchery broodstock consists of both hatchery- and wild-born adults. 1, 1, x are the relative fitness at each locus of the genotypes in the wild. Selection occurs at 5 loci.

APPENDIX I

Introduction

What follows is a brief description of the computer model, a guide to reading Pascal programs, a table of contents of the various sections, procedures, and functions of the model for ease of reference and to provide a general overview of the model, and finally, the program code itself.

Note that minor changes--syntax or otherwise--to the code have been made in the interest of clarity. This means as presented the source code won't compile in TurboPascal 7.0. However the few changes necessary to make the code compile are indicated in specific line comments. Anybody who wants a copy of the program should send the author a 3.5" disk and a SASE.

Description of model

The simulation model was written in Pascal (Turbo Pascal ver. 7.0), and in short, the program does the following:

(1) Generates a table of all 243 possible genotypes (5 loci and 3 genotypes for each loci, equals 3^5 possible genotypes), their fitness and their expected Hardy Weinberg Proportions.

(2) Randomly samples from all 243 genotypes to create an initial population of N (= 2000) individuals.

(3) Randomly divides the population into 2 parent populations. A wild parent population of size βN which mates in the wild and a captive parent population of size $(1 - \beta)N$ which mates in the hatchery.

Then for both Wild and Captive Populations

(4) Randomly picks a male and female to mate.

(5) Creates a zygote and determines whether a zygote survives or not by comparing a random number with the zygote's fitness. A zygote's fitness is a function of which environment--wild or captive--it is in. Fitness of individuals with multi-locus genotypes are determined by multiplying the fitness of each genotype at each locus together. See Table 1 for fitness values in the different environments.

(6) For each population, steps 4 and 5 are repeated until βN or $(1 - \beta)N$ individuals are created for the next generation.

(7) Once all individuals from both populations have been created, regroups all individuals and calculates fitness of the population as a wild population.

(8) Steps 3 - 7 are repeated

Reading Pascal programs

For those readers unfamiliar with Pascal language, Pascal is a structured language and programs written in Pascal contain three main parts:

(1) a variable section where units, data types, global variables and constants are defined,
(2) a procedure and function section where the details (subroutines) of the program are carried out, and
(3) the main body of the program which shows the overall structure of the program.

Programs are read from the bottom up. That is, the "program" is at the end of the code, and "calls" procedures or functions that precede the program block. The variable section precedes the procedure and function section. Also note that within procedures and functions local variables and constants, or calls to other functions or procedures are allowed and occur syntactically before the body of the procedure or function. For specific and detailed description on how Pascal work see any of the books on Pascal or Borland TurboPascal manuals. I found the book *Oh! Pascal* (Cooper and Clancy 1982) particularly helpful. For scientific computing see particularly the book *Numerical Recipes for Pascal: The Art of Scientific Computing* (Press et al. 1989).

Table of contents
Supplementation Model Program Code

Conventions

- *Comments that refer to a procedure or block of code are italicized text and are set off by a lined border at the beginning and end of each comment section, and precede the code itself.*

- Comments that refer to a specific line of code are *italicized* text and set off by {...}after the line of code.

- Constants, variables, and data types are unformatted text. Variable names longer than one word are connected by an underscore, e.g., Returning_Adult_Population_Size. Global variable have each beginning letter capitalized, e.g., Male_Gamete. Local variables are all lowercase, e.g., random_integer.

- Procedures and functions names are ***italic bold*** text. Procedure and function names longer than one word are strung together with capital letters separating word, e.g., ***HatcheryZygoteFitness***. Procedures and functions names are only italic bold text in the procedures and function section of the program; for sake of clarity in the program block they are unformatted text.

- Reserved words in Pascal are **lowercase bold** text, e.g. **begin.**

- Groups of procedures and functions that can be conceptually organized into units are set off by a single underlined line with centered **bold** text.

- Tabs are used to indicate logic (nesting) of for loops, while loops, and if-then and case statements

Supplementation model

```
program Supplementation
{$N+}

uses DOS;

{"W" alleles are favored in the wild. "H" alleles are favored in the hatchery. The Arabic numerals refer to the locus, the alleles are located at. The values are the initial frequencies of the alleles in the population.}
const
Frequency_W1: real = 0.95; {allele frequency of W1 allele}
Frequency_H1: real = 0.05; {allele frequency of H1 allele}

Frequency_W2: real = 0.95; {allele frequency of W2 allele}
```

Frequency_H2: real = 0.05; {*allele frequency of H2 allele*}

Frequency_W3: real = 0.95; {*allele frequency of W3 allele*}
Frequency_H3: real = 0.05; {*allele frequency of H3 allele*}

Frequency_W4: real = 0.95; {*allele frequency of W4 allele*}
Frequency_H4: real = 0.05; {*allele frequency of H4 allele*}

Frequency_W5: real = 0.95; {*allele frequency of W5 allele*}
Frequency_H5: real = 0.05; {*allele frequency of H5 allele*}

Fitness set for wild zygotes and returning adults--once set a constant throughout a simulation. See Table 1.

Fitness_Wild_W1W1: real = 1.0;
Fitness_Wild_W1H1: real = 1-sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Wild_H1H1: real = s; {*s = 0.75, 0.50, or 0.25*}

Fitness_Wild_W2W2: real = 1.0;
Fitness_Wild_W2H2: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Wild_H2H2: real = s;{*s = 0.75, 0.50, or 0.25*}

Fitness_Wild_W3W3: real = 1.0;
Fitness_Wild_W3H3: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Wild_H3H3: real = s;{*s = 0.75, 0.50, or 0.25*}

Fitness_Wild_W4W4: real = 1.0;
Fitness_Wild_W4H4: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Wild_H4H4: real = s;{*s = 0.75, 0.50, or 0.25*}

Fitness_Wild_W5W5: real = 1.0;
Fitness_Wild_W5H5: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Wild_H5H5: real = s;{*s = 0.75, 0.50, or 0.25*}

Fitness set for hatchery zygotes--once set a constant throughout the simulation

---

Fitness_Hatchery_W1W1: real = s;{*s = 0.75, 0.50, or 0.25*}
Fitness_Hatchery_W1H1: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Hatchery_H1H1: real = 1.0;

Fitness_Hatchery_W2W2: real = s;{*s = 0.75, 0.50, or 0.25*}
Fitness_Hatchery_W2H2: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Hatchery_H2H2: real = 1.0;

Fitness_Hatchery_W3W3: real = s;{*s = 0.75, 0.50, or 0.25*}
Fitness_Hatchery_W3H3: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}

Fitness_Hatchery_H3H3: real = 1.0;

Fitness_Hatchery_W4W4: real = s;{*s = 0.75, 0.50, or 0.25*}
Fitness_Hatchery_W4H4: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Hatchery_H4H4: real = 1.0;

Fitness_Hatchery_W5W5: real = s;{*s = 0.75, 0.50, or 0.25*}
Fitness_Hatchery_W5H5: real = 1 - sz; {*s = 0.75, 0.50, or 0.25; z = 0 or 0.5*}
Fitness_Hatchery_H5H5: real = 1.0;

Probability_Wild_Bound: real = β; {β = *0.50 or 0.75; Probability that an individuals will mate in the wild*}
Returning_Adult_Population_Size: integer = N; {*2000*}
Wild_Zygote_Population_Size: integer = βN {*1000 or 1500*};
Hatchery_Population_Size: integer = (1 - β)N {*1000 or 500*};

Data types used in the model

```
type
TAlleles = (W1, H1, W2, H2, W3, H3, W4, H4, W5, H5);
TAlleleRecord = record
    Allele: array [1..5, 1..2] of TAlleles
    end;
TGamete = record
    Allele: array [1..5] of TAlleles
    end;
TGenotypeRecord = record
    Allele: array [1..5, 1..2] of TAlleles;
    HardyWeinberg: real;
    end;
TGenotypesArray = array [1..243] of TGenotypeRecord;
TBirthPlace = (WildBorn, HatcheryBorn);
TEnvironment = (Wild_Adult, Wild_Zygote, Hatchery);
TIndividual = record
    Time: integer;
    BirthPlace: TBirthPlace;
    Habitat: TEnvironment;
    Genotype: TAlleleRecord;
    end; {end of record individual}
TPopulation = array [1..2000] of TIndividual;
TWildPopulation = array [1..1000] of TIndividual; {or depending on the value of β
array [1..1500] of TIndividual;}
THatcheryPopulation = array [1..1000] of TIndividual; {or depending of the value of β
array [1..500] of TIndividual;}
Table = array [1..97] of real;
```

Global variables used in the model

```
var
Genotype: TGenotypeRecord;
Possible_Genotypes: TGenotypesArray;
Genotype_Counter: integer;
Returning_Adult_Population: TPopulation;
Number_of_Runs: integer;
Number_of_Replicates: integer;
Genotype_Picked: integer;
Wild_Zygote_Population: TWildPopulation;
Hatchery_Population: THatcheryPopulation;
Male, Female: TIndividual;
Male_Gamete, Female_Gamete: TGamete;
Zygote: TAlleleRecord;
Zygote_Counter: integer;
Count_W1, Count_W2, Count_W3, Count_W4, Count_W5: integer;
Count_H1, Count_H2, Count_H3, Count_H4, Count_H5: integer;
Allele_Frequency_W1, Allele_Frequency_W2, Allele_Frequency_W3,
Allele_Frequency_W4, Allele_Frequency_W5: real;
Allele_Frequency_H1, Allele_Frequency_H2, Allele_Frequency_H3,
Allele_Frequency_H4, Allele_Frequency_H5: real;
FreqFile: text;
Mean_Fitness_Returning_Population: real;
FitnessFile: text;
Initial_Seed: longint;
Modulus1, Modulus2, Modulus3: Longint;
Multiplier1, Multiplier2, Multiplier3: longint;
Increment1, Increment2, Increment3: longint;
Seed1, Seed2, Seed3: longint;
Random_Shuffle_Table: Table;
Random_Number: real;
```

Random Number And Shuffling Procedures

*The procedure **Seed** uses 3 linear congruential generators and a shuffle routine to break up sequential correlations in linear congruential generators. The procedure provides a seed value for the random number generator and initializes a table of random numbers which is then used in procedure **RandomNumberGenerator** to shuffle random numbers. Design of random number generator and choice of values of Modulus, Increment and Multiplier from* Numerical Recipes in Pascal: The Art of Scientific Computing *(Press et al. 1989)*

```
procedure Seed; { Seeds RandomNumberGenerator}
var
```

```
i, j: integer; {counters}
begin
Modulus1:= 259200;
Multiplier1:= 7141;
Increment1:= 54773;
Modulus2:= 134456;
Multiplier2:= 8121;
Increment2:= 28411;
Modulus3:= 243000;
Multiplier3:= 4561;
Increment3:= 51349;
Seed1:= (Increment1+Initial_Seed)mod Modulus1;
Seed1:= ((Multiplier1*Seed1)+Increment1) mod Modulus1;
Seed2:= Seed1 mod Modulus2;
Seed3:= Seed1 mod Modulus3;
For j:= 1 to 97 do begin
   Seed1:= ((Multiplier1*Seed1)+Increment1) mod Modulus1;
   Random_Shuffle_Table[j]:= (Seed1+(Seed2/Modulus2))/Modulus1;
   end; {end for loop}
end; {end begin loop}
```

---

*The procedure **RandomNumberGenerator** uses the seed value and shuffle table from procedure **Seed** to generate a random number that has a "period which for all practical purposes is infinite" (Press et al. 1989). Random numbers are drawn from the shuffle table which generates a "uniform" random number between 0 and 1. (see comments for procedure **Seed**)*

```
procedure RandomNumberGenerator;
var
i: integer;
begin
Seed1:= ((Multiplier1*Seed1)+Increment1) mod Modulus1;
Seed2:= ((Multiplier2*Seed2)+Increment2) mod Modulus2;
Seed3:= ((Multiplier3*Seed3)+Increment3) mod Modulus3;
repeat i:= 1 + Trunc((97*Seed3)/Modulus3);
until (i >= 1) and (i <= 97);
Random_Number:= Random_Shuffle_Table[i];
Random_Shuffle_Table[i]:= (Seed1+(Seed2/Modulus2))/Modulus1;
end;
```

---

*The procedure **ShuffleReturningAdultPopulation** is used to shuffle the Returning_Adult_Population after the second run since the Returning_Adult_Population is sequentially written. The first βN individuals of the Returning_Adult_Population come from the surviving wild zygotes and the last (1 - β)N individuals come from surviving hatchery zygotes. The shuffling of the Returning_Adult_Population is used to avoid*

biasing the selection of wild or hatchery bound fish which can possibly occur because of the sequential writing of wild and Hatchery bound individuals to the array. That is, since βN wild individuals might possibly be selected before all (1 - β)N hatchery individuals have been selected, the Returning_Adult_Population is shuffled to avoid always sending the tail-end of the Returning_Adult_Population--that is the hatchery zygote population portion off the Returning_Adult_Population always to one population or another.

```
procedure ShuffleReturningAdultPopulation;
var
i: integer;
temporary: TPopulation;
random_integer: Integer;
begin
For i:= 1 to Returning_Adult_Population_Size do begin
   RandomNumberGenerator;
   random_integer:= 1+trunc(Returning_Adult_Population_Size*Random_Number);
   temporary[i]:= Returning_Adult_Population[i];
   Returning_Adult_Population[i]:= Returning_Adult_Population[random_integer];
   Returning_Adult_Population[random_integer]:= temporary[i];
   end; {for end}
end; {procedure end}
```

Mating, Gametogenesis, & Zygote Viability Selection Procedures

The procedure NextGeneration writes the surviving wild and hatchery zygotes to the Returning_Adult_Population. The previous Returning_Adult_Population is written over and time is advance +1 unit. The Returning_Adult_Population is written sequentially with the first βN individuals coming from the wild and the next (1 - β)N individuals from the hatchery.

```
procedure NextGeneration;
var
i, j: integer;
begin
with Returning_Adult_Population[Zygote_Counter] do begin
Time:= Time + 1;
if Zygote_Counter <= Wild_Zygote_Population_Size then begin
   BirthPlace:= WildBorn;
   end
else begin
   BirthPlace:= HatcheryBorn;
   end;
Habitat:= Wild_Adult;
For i:= 1 to 5 do begin
   For j:= 1 to 2 do begin
```

```
      Genotype.Allele[i,j]:= Zygote.Allele[i,j];
      end; {for j end}
   end; {for i end}
end; {with end}
Zygote_Counter:= Zygote_Counter + 1;
end; {procedure end}
```

---

*The function **HatcheryZygoteFitness** calculates the fitness of a hatchery zygote's genotype called by the procedure **HatcheryZygoteSurvival***

```
Function HatcheryZygoteFitness : real;
var
hatchery_zygote_fitness_1, hatchery_zygote_fitness_2, hatchery_zygote_fitness_3,
hatchery_zygote_fitness_4, hatchery_zygote_fitness_5: real;
i: integer; {counter}
begin
For i:= 1 to 5 do begin
   case (ord(Zygote.Allele[i,1])+ord(Zygote.Allele[i,2])) of
      0: hatchery_zygote_fitness_1:= Fitness_Hatchery_W1W1;
      1: hatchery_zygote_fitness_1:= Fitness_Hatchery_W1H1;
      2: hatchery_zygote_fitness_1:= Fitness_Hatchery_H1H1;

      4: hatchery_zygote_fitness_2:= Fitness_Hatchery_W2W2;
      5: hatchery_zygote_fitness_2:= Fitness_Hatchery_W2H2;
      6: hatchery_zygote_fitness_2:= Fitness_Hatchery_H2H2;

      8: hatchery_zygote_fitness_3:= Fitness_Hatchery_W3W3;
      9: hatchery_zygote_fitness_3:= Fitness_Hatchery_W3H3;
      10: hatchery_zygote_fitness_3:= Fitness_Hatchery_H3H3;

      12: hatchery_zygote_fitness_4:= Fitness_Hatchery_W4W4;
      13: hatchery_zygote_fitness_4:= Fitness_Hatchery_W4H4;
      14: hatchery_zygote_fitness_4:= Fitness_Hatchery_H4H4;

      16: hatchery_zygote_fitness_5:= Fitness_Hatchery_W5W5;
      17: hatchery_zygote_fitness_5:= Fitness_Hatchery_W5H5;
      18: hatchery_zygote_fitness_5:= Fitness_Hatchery_H5H5;
      end {end case}
   end; {end for loop}
HatcheryZygoteFitness:= (hatchery_zygote_fitness_1) * (hatchery_zygote_fitness_2) *
(hatchery_zygote_fitness_3) * (hatchery_zygote_fitness_4) *
(hatchery_zygote_fitness_5);
end; {end Fitness Function}
```

*The procedure **HatcheryZygoteSurvival** determines the survival of the hatchery zygote by drawing a random number and comparing it with the fitness of the zygote. If the zygote survives then procedure **NextGeneration** is called. Hatchery zygotes are written as the (βN)+1 to N individuals of the Returning_Adult_Population. If the zygote doesn't survive then control is passed back to the main program.*

```
procedure HatcheryZygoteSurvival;
begin
HatcheryZygoteFitness;
RandomNumberGenerator;
If Random_Number ≤ HatcheryZygoteFitness then begin
   NextGeneration;
   end {end if begin}
end; {end procedure}
```

*The function **WildZygoteFitness** calculates the fitness of a wild zygote's genotype and is called by procedure **WildZygoteSurvival**.*

```
Function WildZygoteFitness: real;
var
wild_zygote_fitness_1, wild_zygote_fitness_2, wild_zygote_fitness_3,
wild_zygote_fitness_4, wild_zygote_fitness_5: real;
i: integer; {counter}
begin
For i:= 1 to 5 do begin
   case (ord(Zygote.Allele[i,1])+ord(Zygote.Allele[i,2])) of
      0: wild_zygote_fitness_1:= Fitness_Wild_W1W1;
      1: wild_zygote_fitness_1:= Fitness_Wild_W1H1;
      2: wild_zygote_fitness_1:= Fitness_Wild_H1H1;

      4: wild_zygote_fitness_2:= Fitness_Wild_W2W2;
      5: wild_zygote_fitness_2:= Fitness_Wild_W2H2;
      6: wild_zygote_fitness_2:= Fitness_Wild_H2H2;

      8: wild_zygote_fitness_3:= Fitness_Wild_W3W3;
      9: wild_zygote_fitness_3:= Fitness_Wild_W3H3;
      10: wild_zygote_fitness_3:= Fitness_Wild_H3H3;

      12: wild_zygote_fitness_4:= Fitness_Wild_W4W4;
      13: wild_zygote_fitness_4:= Fitness_Wild_W4H4;
      14: wild_zygote_fitness_4:= Fitness_Wild_H4H4;

      16: wild_zygote_fitness_5:= Fitness_Wild_W5W5;
      17: wild_zygote_fitness_5:= Fitness_Wild_W5H5;
      18: wild_zygote_fitness_5:= Fitness_Wild_H5H5;
```

```
      end {end case statement}
   end; {end for loop}
WildZygoteFitness:= (wild_zygote_fitness_1) * (wild_zygote_fitness_2) *
(wild_zygote_fitness_3) * (wild_zygote_fitness_4) * (wild_zygote_fitness_5)
end; {end function}
```

*The procedure **WildZygoteSurvival** determines the survival of a wild zygote by drawing a random number and comparing it with the fitness of the zygote. If the zygote survives then procedure **NextGeneration** is called. Wild zygotes are written as the first βN individuals of the Returning_Adult_Population. If zygote doesn't survive then nothing happens and control is passed back to the main program.*

```
procedure WildZygoteSurvival;
var
i, j: integer;
begin
WildZygoteFitness;
RandomNumberGenerator;
if Zygote_Counter ≤Wild_Zygote_Population_Size then begin
   if Random_Number ≤ WildZygoteFitness then begin
      NextGeneration;
      end {end random number if }
   end; {end zygote counter if}
end; {end procedure}
```

*The procedure **ZygoteFormation** forms the zygote by combining the female and male gametes together. This procedure is used both by wild zygote and hatchery populations.*

```
procedure ZygoteFormation;
var
i: integer;
begin
For i:= 1 to 5 do begin
   Zygote.Allele[i,1]:= Female_Gamete.Allele[i];
   Zygote.Allele[i,2]:= Male_Gamete.Allele[i];
   end; {end for loop}
end; {end procedure}
```

*The procedure **FemaleGametogenesis** determines which alleles a female gives to her offspring. If the locus in question is heterozygous then the procedure draws a random number and compares it with the probability of getting Allele1, i.e., the allele in the first position in the 1x2 genotype matrix, and makes a decision. If the locus is homozygous then it uses Allele1 for the egg. The procedure uses the functions odd and ord of alleles to determine heterozygosity. The procedure is used both by wild zygote and hatchery populations.*

```
procedure FemaleGametogenesis;
const
probabilityallele1: real = 0.5;
var
i: integer;
begin
For i:= 1 to 5 do begin
   if odd((ord(Female.Genotype.Allele[i,1])+ord(Female.Genotype.Allele[i,2]))) then
   begin
      RandomNumberGenerator;
      if Random_Number <= probabilityallele1 then begin
         Female_Gamete.Allele[i]:= Female.Genotype.Allele[i,1];
         end
      else begin
         Female_Gamete.Allele[i]:= Female.Genotype.Allele[i,2];
         end;
      end {end if odd...then begin}
   else begin
      Female_Gamete.Allele[i]:= Female.Genotype.Allele[i,1];
      end;
   end; {For end}
end; {end procedure}
```

---

*The procedure **MaleGametogenesis** determines which alleles a males gives to his offspring. If the locus in question is heterozygous then a random number is drawn and compared with probability of getting Allele1, i.e., the first allele in the 1x2 genotype matrix, and a decision is made. If it is homozygous then Allele1 is used for sperm. The procedure uses function odd and ord to determine heterozygosity. The procedure is used by both wild zygote and hatchery populations.*

```
procedure MaleGametogenesis;
const
probabilityallele1: real = 0.5;
var
i: integer;
begin
For i:= 1 to 5 do begin
   if odd((ord(Male.Genotype.Allele[i,1])+ord(Male.Genotype.Allele[i,2]))) then begin
      RandomNumberGenerator;
      if Random_Number <= probabilityallele1 then begin
         Male_Gamete.Allele[i]:= Male.Genotype.Allele[i,1];
         end
      else begin
         Male_Gamete.Allele[i]:= Male.Genotype.Allele[i,2];
         end;
```

```
      end {if odd then begin}
   else begin
      Male_Gamete.Allele[i]:= Male.Genotype.Allele[i,1];
      end;
   end; {for end}
end; {end procedure}
```

---

*The procedures **PickWildFemale** and **PickHatcheryFemale** define the first half of a population, i.e., (βN)/2 or ((1 - β)N)/2 individuals, as female. A female is picked for mating by drawing a random number between 0 and 1, converting it to a random integer between 1 and (βN)/2 or ((1 - β)N)/2 as the case may be, and assigning the population's record id which corresponds to the random number as a female who mates. The female is then used in procedure **FemaleGametogenesis** to determine which genes she will pass to her offspring.*

```
procedure PickWildFemale (Population: TWildPopulation);
var
random_integer: integer;
begin
RandomNumberGenerator;
random_integer:= 1+trunc(0.5*Wild_Zygote_Population_Size*Random_Number);
Female:= Population[random_integer];
end;

procedure PickHatcheryFemale (Population: THatcheryPopulation);
var
random_integer: integer;
begin
RandomNumberGenerator;
random_integer:= 1+trunc(0.5*Hatchery_Population_Size*Random_Number);
Female:= Population[random_integer];
end;
```

---

*The procedure **PickMale** defines the second half of a population, i.e., (βN)/2 or ((1 - β)N)/2 individuals, as males. A male is picked for mating by drawing a random number between 0 and 1, converting it to a random integer between (βN)/2 +1 to (βN) or ((1 - β)N)/2 +1 to ((1 - β)N) as the case may be, and assigning the population's record id which corresponds to the random number as a male who will mate with a chosen female. The male is then used in procedure **MaleGametogenesis** to determine which genes the father will give to his offspring. The procedure is used by both wild zygote and hatchery populations.*

```
procedure PickWildMale (Population: TWildPopulation);
var
random_integer: integer;
```

```
begin
RandomNumberGenerator;
random_integer:= 1 + trunc(0.5 * Wild_Zygote_Population_Size)+trunc(0.5 *
Wild_Zygote_Population_Size * Random_Number);
Male:= Population[random_integer];
end;

procedure PickHatcheryMale (Population: THatcheryPopulation);
var
random_integer: integer;
begin
RandomNumberGenerator;
random_integer:= 1+trunc(0.5 * Hatchery_Population_Size)+trunc(0.5 *
Hatchery_Population_Size * Random_Number);
Male:= Population[random_integer];
end;
```

Calculate Returning Adult Fitness Procedures

The procedure CountAlleles counts the number of alleles in the population. The procedure uses case, and ord functions to count alleles.

```
procedure CountAlleles;
var
i, j, k: integer;
begin
Count_W1:= 0;
Count_W2:= 0;
Count_W3:= 0;
Count_W4:= 0;
Count_W5:= 0;
Count_H1:= 0;
Count_H2:= 0;
Count_H3:= 0;
Count_H4:= 0;
Count_H5:= 0;
For k:= 1 to 2000 do begin
   For i:= 1 to 5 do begin
      case (ord (Returning_Adult_Population[k].Genotype.Allele[i,1]) +
      ord(Returning_Adult_Population[k].Genotype.Allele[i,2])) of
         0: Count_W1:= Count_W1+2;
         1: begin
            Count_W1:= Count_W1+1;
            Count_H1:= Count_H1+1;
            end;
```

```
            2: Count_H1:= Count_H1+2;
            4: Count_W2:= Count_W2+2;
            5: begin
               Count_W2:= Count_W2+1;
               Count_H2:= Count_H2+1;
               end;
            6: Count_H2:= Count_H2+2;
            8: Count_W3:= Count_W3+2;
            9: begin
               Count_W3:= Count_W3+1;
               Count_H3:= Count_H3+1;
               end;
           10: Count_H3:= Count_H3+2;
           12: Count_W4:= Count_W4+2;
           13: begin
               Count_W4:= Count_W4+1;
               Count_H4:= Count_H4+1;
               end;
           14: Count_H4:= Count_H4+2;
           16: Count_W5:= Count_W5+2;
           17: begin
               Count_W5:= Count_W5+1;
               Count_H5:= Count_H5+1;
               end;
           18: Count_H5:= Count_H5+2;
           end{case end}
       end; {end for i}
   end; {end for k}
end; {end procedure}
```

---

*The procedure **CalculateAlleleFrequencies** uses count data from the procedure **CountAlleles** to calculate the frequency of the alleles in the Returning_Adult_Population.*

```
procedure CalculateAlleleFrequencies;
begin
Allele_Frequency_W1:= Count_W1/4000;
AlleleFreqC1:= Count_H1/4000;
Allele_Frequency_W2:= Count_W2/4000;
AlleleFreqC2:= Count_H2/4000;
Allele_Frequency_W3:= Count_W3/4000;
AlleleFreqC3:= Count_H3/4000;
Allele_Frequency_W4:= Count_W4/4000;
AlleleFreqC4:= Count_H4/4000;
Allele_Frequency_W5:= Count_W5/4000;
```

```
AlleleFreqC5:= Count_H5/4000;
end;
```

---

*The procedure **WriteAlleleFrequencies** writes each generations allele frequency data to the screen.*

```
procedure WriteAlleleFrequencies;
var
replicate: integer;
run: integer;
time: integer;
begin
replicate:= Number_of_Replicates;
run:= Number_of_Runs;
time:= Returning_Adult_Population[1].Time;
write ('Rep':5, 'Run':5, 'Time':5, 'p(W1)':6,'p(H1)':6,'p(W2)':6,'p(H2)':6, 'p(W3)':6,
'p(H3)':6, 'p(W4)':6);
writeln ('p(H4)':7,'p(W5)':7, 'p(H5)':7);
write (replicate:5, run:5, time:5, Allele_Frequency_W1:6:3, Allele_Frequency_H1:6:3,
Allele_Frequency_W2:6:3, Allele_Frequency_H2:6:3, Allele_Frequency_W3:7:3);
writeln (Allele_Frequency_H3:6:3, Allele_Frequency_W4:6:3, Allele_Frequency_H4:6:3,
Allele_Frequency_W5:6:3, Allele_Frequency_H5:6:3);
end;
```

---

*The procedure **WriteHeaderFileAlleleFrequencies** writes a header of text to a file. The purpose of the header is identify files.*

```
procedure WriteHeaderFileAlleleFrequencies;
var
replicate: integer;
run: integer;
time: integer;
begin
replicate:= Number_of_Replicates;
run:= Number_of_Runs;
time:= Returning_Adult_Population[1].Time;
write (FreqFile,'Rep':5, 'Run':5, 'Time':5, 'p(W1)':6,'p(H1)':6,'p(W2)':6,'p(H2)':6,
'p(W3)':6, 'p(H3)':6, 'p(W4)':6);
writeln (FreqFile,'p(H4)':7,'p(W5)':7, 'p(H5)':7);
end;
```

---

*The procedure **WriteFileAlleleFrequencies** writes each generations allele frequency data to a file.*

```
procedure WriteFileAlleleFrequencies;
var
```

```
replicate: integer;
run: integer;
time: integer;
begin
replicate:= Number_of_Replicates;
run:= Number_of_Runs;
time:= Returning_Adult_Population[1].Time;
write (FreqFile, Replicate:5, Run:5, Time:5, Allele_Frequency_W1:6:3,
Allele_Frequency_H1:6:3, Allele_Frequency_W2:6:3, Allele_Frequency_H2:6:3);
writeln (FreqFile, Allele_Frequency_W3:6:3, Allele_Frequency_H3:6:3,
Allele_Frequency_W4:6:3, Allele_Frequency_H4:6:3, Allele_Frequency_W5:6:3,
Allele_Frequency_H5:6:3);
end;
```

---

*The function **ReturningAdultFitness** calculate the fitness of each returning adult. The fitness set used is the same as the fitness set for wild zygotes (see Table 1). The function is called by procedure **AverageFitnessReturningAdultPopulation**;*

```
function ReturningAdultFitness (ReturningAdultIndividual: TIndividual): real;
var
   returning_adult_fitness1, returning_adult_fitness2,
   returning_adult_fitness3,returning_adult_fitness4, returning_adult_fitness5: real;
   i: integer; {counter}
begin
For i:= 1 to 5 do begin
   case (ord(Returning_Adult_Individual.Genotype.Allele[i,1]) +
   ord(Returning_Adult_Individual.Genotype.Allele[i,2])) of
      0: returning_adult_fitness1:= Fitness_Wild_W1W1;
      1: returning_adult_fitness1:= Fitness_Wild_W1H1;
      2: returning_adult_fitness1:= Fitness_Wild_H1H1;
      4: returning_adult_fitness2:= Fitness_Wild_W2W2;
      5: returning_adult_fitness2:= Fitness_Wild_W2H2;
      6: returning_adult_fitness2:= Fitness_Wild_H2H2;
      8: returning_adult_fitness3:= Fitness_Wild_W3W3;
      9: returning_adult_fitness3:= Fitness_Wild_W3H3;
      10: returning_adult_fitness3:= Fitness_Wild_H3H3;
      12: returning_adult_fitness4:= Fitness_Wild_W4W4;
      13: returning_adult_fitness4:= Fitness_Wild_W4H4;
      14: returning_adult_fitness4:= Fitness_Wild_H4H4;
      16: returning_adult_fitness5:= Fitness_Wild_W5W5;
      17: returning_adult_fitness5:= Fitness_Wild_W5H5;
      18: returning_adult_fitness5:= Fitness_Wild_H5H5;
      end {end case}
   end; {end for loop}
```

```
ReturningAdultFitness:= (returning_adult_fitness1) * (returning_adult_fitness2) *
(returning_adult_fitness3) * (returning_adult_fitness4) * (returning_adult_fitness5)
end; {end procedure}
```

---

*The procedure **MeanFitnessReturningAdultPopulation** calculates the mean fitness of the Returning_Adult_Population by summing the fitness for each individual and dividing by N (=2000) The output of the model.*

```
procedure MeanFitnessReturningAdultPopulation;
type
TReturningAdultFitnessRecord = record
   Fitness: real;
   end;
TReturningAdultFitnessArray = array [1..2000] of TReturningAdultFitnessRecord;
var
i: integer;
returning_adult_individual_fitness: TReturningAdultFitnessArray;
j: integer;
sum_returning_adult_fitness: real;
begin
For i:= 1 to Returning_Adult_Population_Size do begin
   returning_adult_individual_fitness[i].fitness:=
   ReturningAdultFitness(Returning_Adult_Population[i]);
   end; {end for loop}
sum_returning_adult_fitness:= 0.0;
For j:= 1 to Returning_Adult_Population_Size do begin
   sum_returning_adult_fitness:= sum_returning_adult_fitness +
   returning_adult_individual_fitness[j].fitness;
   end; {end for loop}
Mean_Fitness_Returning_Population:= (sum_returning_adult_fitness) /
returning_adult_population_size;
end; {end procedure}
```

---

*The procedure **FitnessToScreen** writes the calculated average fitness of the Returning_Adult_Population to the screen.*

```
procedure FitnessToScreen (Mean_Fitness_Returning_Population: real);
var
replicate: integer;
run: integer;
time1: integer;
begin
replicate:= Number_of_Replicates;
run:= Number_of_Runs;
time1:= Returning_Adult_Population[1].Time - 1;
writeln (replicate:4, run:4, time1:5, Mean_Fitness_Returning_Population:11:8);
```

```
writeln; {adds a line break between generations}
end; {end procedure}
```

*The procedure **HeaderFitnessToFile** writes a header of text to the file that stores each generations fitness data. The purpose of the procedure is to identify the file.*

```
procedure HeaderFitnessToFile;
begin
writeln (FitnessFile,' ':4,' ':4,' ':5,'Fitness':11);
writeln (FitnessFile,' ':4,' ':4,' ':5,'Returning_Adult_':11,);
writeln (FitnessFile,'Rep':4,'Run':4,'Time':5,'Pop':11);
writeln (FitnessFile,'--------------------------------------------------------------------------------');
end;
```

The procedure FitnessToFile writes each generations mean fitness of the Returning_Adult_Population to a file. The model's output.

```
procedure FitnessToFile (Mean_Fitness_Returning_Population: real);
var
replicate: integer;
run: integer;
time1: integer;
begin
replicate:= Number_of_Replicates;
run:= Number_of_Runs;
time1:= Returning_Adult_Population[1].Time - 1;
writeln (FitnessFile, replicate:4, run:4, time1:5,
Mean_Fitness_Returning_Population:11:8);
writeln;
end;
```

Wild Or Hatchery Bound Procedures

*The procedure **WildorHatcheryBound** determines by going sequentially through the Returning_Adult_Population array, drawing a random number and making a decision whether an individual of the Returning_Adult_Population is chosen to mate in the wild or captivity. The procedure also creates two parent populations a Wild_Zygote_Population of size βN individuals and a Hatchery_Population of size $(1 - \beta)N$ individuals. Code for two procedures are listed. The first one is the code used if wild and hatchery born returning adults have equal probability of mating in the hatchery. The second procedure is the code used in simulations where only wild born returning adults spawn in the hatchery.*

```
procedure WildorHatcheryBound; {procedure used in wild- and hatchery-born
returning adults have equal probability of mating in the hatchery}
var
```

```
returning_adult_counter: integer;
wild_counter: integer;
hatchery_counter: integer;
begin
wild_counter:= 1;
hatchery_counter:= 1;
For returning_adult_counter:= 1 to Returning_Adult_Population_Size do begin
   if (wild_counter <= Wild_Zygote_Population_Size) and (hatchery_counter <=
   Hatchery_Population_Size) then begin
      RandomNumberGenerator;
      if (Random_Number <= Probability_Wild_Bound) then begin
         Wild_Zygote_Population[wild_counter]:=
            Returning_Adult_Population[returning_adult_counter];
         Wild_Zygote_Population[wild_counter].Habitat:= Wild;
         wild_counter:= wild_counter + 1;
         end {end if then begin of Random Number then}
      else begin
         Hatchery_Population[hatchery_counter]:=
         Returning_Adult_Population[returning_adult_counter];
         Hatchery_Population[hatchery_counter].Habitat:= Hatchery;
         hatchery_counter:= hatchery_counter + 1;
         end {end else }
      end {if then begin if (wild_counter ... }
   else begin {else if (wild_counter...}
      if (wild_counter > Wild_Zygote_Population_Size) and (hatchery_counter <=
      Hatchery_Population_Size) then begin
         Hatchery_Population[hatchery_counter]:=
            Returning_Adult_Population[returning_adult_counter];
         Hatchery_Population[hatchery_counter].Habitat:= Hatchery;
         hatchery_counter:= hatchery_counter + 1;
         end {end if then begin (wild_counter ....}
      else begin {else if (wild_counter }
         if (wild_counter ≤ Wild_Zygote_Population_Size) and (hatchery_counter >
         Hatchery_Population_Size) then begin
            Wild_Zygote_Population[wild_counter]:=
               Returning_Adult_Population[returning_adult_counter];
            Wild_Zygote_Population[wild_counter].Habitat:= Wild;
            wild_counter:= wild_counter + 1;
            end {end begin if (wild_counter <}
         end {else if (wild_counter > }
      end; {else if (wild_counter <}
   end; {end for loop}
end; {end procedure}
```

```
procedure WildorHatcheryBound; {procedure used when only wild individuals go into
the hatchery}
var
returning_adult_counter: integer;
wild_counter: integer;
hatchery_counter: integer;
begin
wild_counter:= 1;
hatchery_counter:= 1;
For returning_adult_counter:= 1 to Returning_Adult_Population_Size do begin
        if (Returning_Adult_Population[Returning_Adult_Counter].Birthplace =
        HatcheryBorn) then begin
        Wild_Zygote_Population[wild_counter]:=
        Returning_Adult_Population[returning_adult_counter];
        Wild_Zygote_Population[wild_counter].Habitat:= Wild_Zygote;
        wild_counter:= wild_counter + 1;
        end
      else begin
        if (wild_counter ≤ Wild_Zygote_Population_Size) and (hatchery_counter ≤
        Hatchery_Population_Size) then begin
          RandomNumberGenerator;
          if (Random_Number ≤ Probability_Wild_Bound) then begin
            Wild_Zygote_Population[wild_counter]:=
            Returning_Adult_Population[Returning_Adult_Counter];
            Wild_Zygote_Population[wild_counter].Habitat:= Wild_Zygote;
            wild_counter:= wild_counter + 1;
            end {end if then begin of Random_Number ≤}
        else begin
          Hatchery_Population[hatchery_counter]:=
          Returning_Adult_Population[Returning_Adult_Counter];
          Hatchery_Population[hatchery_counter].Habitat:= Hatchery;
          hatchery_counter:= hatchery_counter + 1;
          end {end else begin of if Random Number}
        end {if then begin of if(wild_counter ≤ Wild_Population_Size).... }
      else begin{else of if (wild_counter ≤ Wild_Population_Size)...}
        if (wild_counter > Wild_Zygote_Population_Size) and (hatchery_counter ≤
        Hatchery_Population_Size) then begin
          Hatchery_Population[hatchery_counter]:=
          Returning_Adult_Population[Returning_Adult_Counter];
          Hatchery_Population[hatchery_counter].Habitat:= Hatchery;
          hatchery_counter:= hatchery_counter + 1;
          end {end if then begin of if (wild_counter > Wild_Zygote_....}
        else begin{else of if (wild_counter > Wild_Zygote_Population_Size)...}
```

```
            if (wild_counter ≤ Wild_Zygote_Population_Size) and (hatchery_counter >
            Hatchery_Population_Size) then begin
               Wild_Zygote_Population[wild_counter]:=
               Returning_Adult_Population[Returning_Adult_Counter];
               Wild_Zygote_Population[wild_counter].Habitat:= Wild;
               wild_counter:= wild_counter + 1;
               end {end begin of if (wild_counter ≤...) and (hatchery_counter >...}
            end {else if (wild_counter >...) and (hatchery_counter ≤....)}
         end; {else if (wild_counter ≤..) and (hatchery_counter ≤..)}
      end; {else if Returning_Adult_Population[Returning_Adult_Counter].....}
   end; {end for loop}
end; {end procedure}
```

Initial Population Procedures

*The procedure **PickGenotype** draws a random number and then chooses a genotype based on the random number and Hardy-Weinberg equilibrium for an individual. The while loop keeps summing the expected genotypic frequency until a sum of genotypic frequencies is greater than the Random number. The genotype which is then used is the i - 1 genotype term in the sum called by procedure **InitialPopulation** to determine a random sample of genotypes for the initial population.*

```
procedure PickGenotype;
var
i: integer;
genotype_frequency: real;
begin
i:= 1;
genotype_frequency:= 0;
RandomNumberGenerator;
while Random_Number >= genotype_frequency do begin
   genotype_frequency:= genotype_frequency + Possible_Genotypes[i].HardyWeinberg;
   i:= i + 1;
   end; {end while loop}
Genotype_Picked:= i - 1;
end; {end procedure}
```

The procedure InitialPopulation creates and initializes an initial population of N individuals drawn randomly from the expected Hardy Weinberg proportions given the initial allele frequencies.

```
procedure InitialPopulation;
var
i,j: integer;
k: integer;
```

```
begin
For k:= 1 to Returning_Adult_Population_Size do begin
   with Returning_Adult_Population[k] do begin
      Returning_Adult_Population[k].Time:= 0;
      Returning_Adult_Population[k].BirthPlace:= WildBorn;
      Returning_Adult_Population[k].Habitat:= Wild_Adult;
      PickGenotype;
      For i:= 1 to 5 do begin
         For j:= 1 to 2 do begin
            Returning_Adult_Population[k].Genotype.Allele[i,j]:=
               Possible_Genotypes[Genotype_Picked].Allele[i,j];
            end; {For j end}
         end; {For i end}
      end; {with end}
   end; {end For k loop}
end; {procedure end}
```

---

*The function **HardyWeinberg** calculates the expected Hardy Weinberg proportions of genotypes at 5 loci. The function is called by the procedure **PossibleGenotypesArray***

```
function HardyWeinberg: real;
var
i: integer;
frequency_genotype_1, frequency_genotype_2, frequency_genotype_3,
frequency_genotype_4, frequency_genotype_5: real;
begin
For i:= 1 to 5 do begin
   case (ord(Genotype.Allele[i,1])+ord(Genotype.Allele[i,2])) of
      0: frequency_genotype_1:= (Frequency_W1)*(Frequency_W1); {p^2}
      1: frequency_genotype_1:= 2*(Frequency_W1)*(Frequency_H1);{2pq}
      2: frequency_genotype_1:= (Frequency_H1)*(Frequency_H1); {q^2}

      4: frequency_genotype_2:= (Frequency_W2)*(Frequency_W2);
      5: frequency_genotype_2:= 2*(Frequency_W2)*(Frequency_H2);
      6: frequency_genotype_2:= (Frequency_H2)*(Frequency_H2);

      8: frequency_genotype_3:= (Frequency_W3)*(Frequency_W3);
      9: frequency_genotype_3:= 2*(Frequency_W3)*(Frequency_H3);
      10: frequency_genotype_3:= (Frequency_H3)*(Frequency_H3);

      12: frequency_genotype_4:= (Frequency_W4)*(Frequency_W4);
      13: frequency_genotype_4:= 2*(Frequency_W4)*(Frequency_H4);
      14: frequency_genotype_4:= (Frequency_H4)*(Frequency_H4);

      16: frequency_genotype_5:= (Frequency_W5)*(Frequency_W5);
```

```
      17: frequency_genotype_5:= 2*(Frequency_W5)*(Frequency_H5);
      18: frequency_genotype_5:= (Frequency_H5)*(Frequency_H5);
      end {case end}
   end; {For end}
HardyWeinberg:= (frequency_genotype_1) * (frequency_genotype_2) *
(frequency_genotype_3) * (frequency_genotype_4) * (frequency_genotype_5);
end; {end function}
```

The procedure PossibleGenotypesArray writes an array of initial genotypes, and hardyweinberg proportions for all 243 possible genotypes.

```
procedure PossibleGenotypesArray;
var
i, j: integer;
begin
Genotype_Counter:= Genotype_Counter +1;
For i:= 1 to 5 do begin
   For j:= 1 to 2 do begin
      Possible_Genotypes[Genotype_Counter].Allele[i,j]:= Genotype.Allele[i,j]
      end; {end For j loop}
   end; {end For i loop}
Possible_Genotypes[Genotype_Counter].HardyWeinberg:= HardyWeinberg;
end; {end procedure Possible_GenotypesArray}
```

---

*The procedures **PossibleGenotypesLocus1, PossibleGenotypesLocus2, PossibleGenotypesLocus3, PossibleGenotypesLocus4,** and **PossibleGenotypesLocus5** randomly walks through all 243 possible genotypes by nested **For** loops that call procedures and **case** statements and each case statement calls the procedure **PossibleGenotypeArray** to write the possible genotypes to an array.*

```
procedure PossibleGenotypesLocus5;
var
locus5genotypes: integer;
begin
For locus5genotypes:= 1 to 3 do begin
   case locus5genotypes of
      1: begin
         Genotype.Allele[5,1]:= W5;
         Genotype.Allele[5,2]:= W5;
         PossibleGenotypesArray;
         end;
      2: begin
         Genotype.Allele[5,1]:= W5;
         Genotype.Allele[5,2]:= H5;
         PossibleGenotypesArray;
```

```
            end;
        3: begin
            Genotype.Allele[5,1]:= H5;
            Genotype.Allele[5,2]:= H5;
            PossibleGenotypesArray;
            end;
        end {end case}
    end; {end for loop}
end; {end procedure}

procedure PossibleGenotypesLocus4;

var
locus4genotypes: integer;
begin
For locus4genotypes:= 1 to 3 do begin
    case locus4genotypes of
        1: begin
            Genotype.Allele[4,1]:= W4;
            Genotype.Allele[4,2]:= W4;
            PossibleGenotypesLocus5
            end;
        2: begin
            Genotype.Allele[4,1]:= W4;
            Genotype.Allele[4,2]:= H4;
            PossibleGenotypesLocus5
            end;
        3: begin
            Genotype.Allele[4,1]:= H4;
            Genotype.Allele[4,2]:= H4;
            PossibleGenotypesLocus5
            end;
        end {end case}
    end; {end for loop}
end; {end procedure}

procedure PossibleGenotypesLocus3;
var
locus3genotypes: integer;
begin
For locus3genotypes:= 1 to 3 do begin
    case locus3genotypes of
        1: begin
            Genotype.Allele[3,1]:= W3;
```

```
            Genotype.Allele[3,2]:= W3;
            PossibleGenotypesLocus4;
            end;
         2: begin
            Genotype.Allele[3,1]:= W3;
            Genotype.Allele[3,2]:= H3;
            PossibleGenotypesLocus4;
            end;
         3: begin
            Genotype.Allele[3,1]:= H3;
            Genotype.Allele[3,2]:= H3;
            PossibleGenotypesLocus4;
            end;
         end {end case}
      end; {end for loop}
end; {end procedure}

procedure PossibleGenotypesLocus2;
var
locus2genotypes: integer;
begin
For locus2genotypes:= 1 to 3 do begin
   case locus2genotypes of
         1: begin
            Genotype.Allele[2,1]:= W2;
            Genotype.Allele[2,2]:= W2;
            PossibleGenotypesLocus3;
            end;
         2: begin
            Genotype.Allele[2,1]:= W2;
            Genotype.Allele[2,2]:= H2;
            PossibleGenotypesLocus3;
            end;
         3: begin
            Genotype.Allele[2,1]:= H2;
            Genotype.Allele[2,2]:= H2;
            PossibleGenotypesLocus3;
            end;
         end {end case}
      end; {end for loop}
end;{end procedure}
```

```
procedure PossibleGenotypesLocus1;
var
locus1genotypes: integer;
begin
For locus1genotypes:= 1 to 3 do begin
   case locus1genotypes of
      1: begin
         Genotype.Allele[1,1]:= W1;
         Genotype.Allele[1,2]:= W1;
         PossibleGenotypesLocus2;
         end;
      2: begin
         Genotype.Allele[1,1]:= W1;
         Genotype.Allele[1,2]:= H1;
         PossibleGenotypesLocus2;
         end;
      3: begin
         Genotype.Allele[1,1]:= H1;
         Genotype.Allele[1,2]:= H1;
         PossibleGenotypesLocus2;
         end;
      end {end case}
   end; {end for loop}
end; {end procedure}
```

Program Block

Note for sake of clarity calls to procedures are not formatted as bold italic, and appear as plain text.

```
begin {program}
{$M 65520, 0, 655360, $N+}  {Compiler directives (M) set stack and heap size; N use
numeric coprocessor}
Assign (FitnessFile, 'C:\wa25h375.dat');
Rewrite (FitnessFile);
Assign (FreqFile, 'C:\Aa25h375.Dat');
Rewrite (FreqFile);
HeaderFitnessToScreen;
HeaderFitnessToFile;
WriteHeaderFileAlleleFrequencies;
For Number_of_Replicates:= 1 to 1 do begin
   Initial_Seed:= 1 + trunc(100000*random);
   writeln ('Initial Seed = ', Initial_Seed);
   Seed;
```

```
    Genotype_Counter:= 0;
    Zygote_Counter:= 1;
    Possible_GenotypesLocus1;
    InitialPopulation;
    For Number_of_Runs:= 1 to 200 do begin
       MeanFitnessReturningAdultPopulation;
       CountAlleleFrequencies;
       CaculateAlleleFrequencies;
       WriteAlleleFrequencies;
       WriteFileAlleleFrequencies;
       WildorHatcheryBound;
       while Zygote_Counter ≤ Wild_Zygote_Population_Size do begin
          PickWildFemale (WildPopulation);
          PickWildMale (WildPopulation);
          FemaleGametogenesis;
          MaleGametogenesis;
          ZygoteFormation;
          WildZygoteSurvival;
          end;
       while (Zygote_Counter > Wild_Zygote_Population_Size) and (Zygote_Counter ≤
       Returning_Adult_Population_Size) do begin
          PickHatcheryFemale (HatcheryPopulation);
          PickHatcheryMale (HatcheryPopulation);
          FemaleGametogenesis;
          MaleGametogenesis;
          ZygoteFormation;
          HatcheryZygoteSurvival;
          end;
       FitnessToScreen (Mean_Fitness_Returning_Population);
       FitnessToFile (Mean_Fitness_Returning_Population);
       ShuffleReturningAdultPopulation;
       Zygote_Counter:= 1;  {reset zygote counter}
       end; {run for loop}
    end; {end replicate for loop}
close (FitnessFile);
close (FreqFile);
writeln ('end of run');
readln;
end.
```

BIBLIOGRAPHY

Allendorf, F. W. and N. Ryman. 1987. Genetic management of hatchery stocks. Pages 141-159 in N. Ryman and F. Utter, editors. *Population genetics and fisheries management.* University of Washington, Seattle, Washington.

Allendorf, F. W. 1993. Delay of adaptation to captive breeding by equalizing family size. *Conservation Biology* 7:416-419.

Belyaev, D. K. 1979. Destabilizing selection as a factor in domestication. *Journal of Heredity* 70:301-308.

Borlase, S. C., D. A. Loebel, R. Frankham, R. K. Nurthen, D. A. Briscoe, and G. E. Daggard. 1993. Modeling problems in conservation genetics using captive *Drosophila* populations: consequences of equalization of family sizes. *Conservation Biology* 7:122-131.

Chourdhury, B. C. and S. Chowdhury. 1986. Lessons from crocodile reintroduction projects in India. *Indian Forester* 112:881-890.

Clune, T., and D. Dauble. 1991. The Yakima/Klickitat fisheries project: a strategy for supplementation of anadromous salmonids. *Fisheries* 16(5):28-34.

Conway, W. 1988. Can technology aid species preservation? Pages 263-268 in E. O. Wilson, editor. *Biodiversity*. National Academy Press, Washington, D. C.

Cooper, D., and M. Clancy. 1982. *Oh! Pascal.* W. W. Norton & Co. New York.

Crouse, D. T., L. B. Crowder, H. Caswell. 1987. A stage-based population model for loggerhead sea turtles and implications for conservation. *Ecology* 68:1412-1423.

Crow, J. F, and M. Kimura. 1970. *An introduction to population genetics theory.* Burgess Publishing Company, Minneapolis, Minnesota.

Cuenco, M. L., T. W. H. Backman, and P. R. Mundy. 1993. The use of supplementation to aid in natural stock restoration. Pages 269-293 in J. G. Cloud and G. H. Thorgaard, editors. *Genetic conservation of salmonid fishes.* Plenum Press, New York, New York.

de Boer, L. E. M. 1992. Current status of captive breeding programs. Pages 5-16 in H. D. M. Moore, W. V. Holt, and G. M. Mace, editors. *Biotechnology and the conservation of genetic diversity*. Oxford University Press, New York.

Dodd, C. K. Jr. 1988. *Synopsis of the biological data on the loggerhead sea turtle* Caretta caretta *(Linnaeus 1758)*. U. S. Fish and Wildlife Service, Biological Report 88(14).

Dodd, C. K. Jr., and R. A. Seigel. 1991. Relocation, repatriation, and translocation of amphibians and reptiles: are they conservation strategies that work? *Herpetologica* 47:336-350.

Ellis, D. H., S. J. Dobrott, and J. G. Goodwin. 1978. Reintroduction techniques for masked bobwhites. Pages 345-354 in S. A. Temple, editor. *Endangered birds: management techniques for preserving threatened species*. University of Wisconsin Press, Madison, Wisconsin.

Foose, T. J., R. Lande, N. R. Flesness, G. Rabb, and B. Read. 1986. Propagation plans. *Zoo Biology* 5:139-146.

Frankel, O. H., and M. E. Soulé. 1981. *Conservation and evolution*. Cambridge University Press, New York.

Frankham, R., H. Hemmer, O. A. Ryder, E. G. Cothran, M. E. Soulé, N. D. Murray, and M. Snyder. 1986. Selection in captive populations. *Zoo Biology* 5:127-138.

Frankham, R., and D. A. Loebel. Modeling conservation genetics problems using captive *Drosophila* populations. 4. Rapid adaptation to captivity. *Zoo Biology*. In press.

Frissell, C. A. 1993. Topology of extinction and endangerment of native fishes in the Pacific Northwest and California (U. S. A.). *Conservation Biology* 7:342-354.

Griffith, B., J. M. Scott, J. W. Carpenter, C. Reed. 1989. Translocation as a species conservation tool: status and strategy. *Science* 245:477-480.

Hartl, D. L., and A. G. Clark. 1989. *Principles of population genetics*. Second edition. Sinauer Associates, Sunderland, Massachusetts.

Hedrick, P. W., P. R. Brussard, F. W. Allendorf, J. A. Beardmore, and S. Orzack. 1986. Protein variation, fitness, and captive propagation. *Zoo Biology* 5:91-100.

Helle, J. H. 1981. Significance of the stock concept in artificial propagation of salmonids in Alaska. *Canadian Journal of Fisheries and Aquatic Sciences* 38:1665-1671.

Howell, P., K. Jones, D. Scarnecchia, L. Lavoy, W. Kendra, and D. Ortman. 1985. Stock assessment of Columbia River anadromous salmonids. Vol. I: Chinook, coho, chum, and sockeye salmon stock summaries. Final Report, Project No. 83-335, Bonneville Power Administration, Portland, Oregon.

Johnson, R. R. 1990. Release and translocation strategies for the Puerto Rican crested toad, *Peltophryne lemu*. *Endangered Species Update* 8:54-57.

Johnsson, J. L., and M. V. Abrahams. 1991. Interbreeding with domestic strain increases foraging under threat of predation in juvenile steelhead trout (Oncorhynchus mykiss): An experimental study. *Canadian Journal of Fisheries and Aquatic Sciences* 48:243-247.

Jonsson, B., N. Jonsson, and L. P. Hansen. 1991. Differences in life history and migratory behavior between wild and hatchery-reared Atlantic salmon in nature. *Aquaculture* 98:69-78.

King, F. W. 1990. Conservation of crocodilians: the release of captive-reared specimens. *Endangered Species Update* 8:48-51.

Klima, E. F. and J. P. McVey. 1982. Headstarting the Kemp's ridley turtle, *Lepidochelys kempi*. Pages 481-487 in K. A. Bjorndal, editor. *Biology and conservation of sea turtles*. Smithsonian Institution Press, Washington, D. C.

Kohane, M. J., and P. A. Parsons. 1988. Domestication: Evolutionary change under stress. *Evolutionary Biology* 23:31-48.

Lima, S. L., and L. M. Dill. 1990. Behavioral decisions made under the risk of predation: a review and prospectus. *Canadian Journal of Zoology* 68:619-640.

Lyles, A. M., and R. M. May. 1987. Problems in leaving the ark. *Nature* 326:245-246.

Marsden, J. E., C. C. Krueger, P. M. Grewe, H. L. Kincaid, B. May. 1993. Genetic comparison of naturally spawned and artificially propagated Lake Ontario lake trout fry: evaluation of a stocking strategy for species rehabilitation. *North American Journal of Fisheries Management* 13:304-317.

Miller, B. J., S. H. Anderson, M. W. DonCarlos, E. T. Thorne. 1988. Biology of the endangered black-ferret and the role of captive propagation in its conservation. *Canadian Journal of Zoology* 66:765-773.

Myers, J. H., and M. D. Sabbath. 1980. Genetic and phenotypic variability, genetic variance and the success of establishment of insect introductions for the biological control of weeds. Pages 91-102 in *Proceedings of the Fifth International Symposium on the Biological Control of Weeds*, Brisbane, Australia.

Nehlsen, W., J. E. Williams, and J. A. Lichatowich. 1991. Pacific salmon at the crossroads: stocks at risk from California, Oregon, Idaho, and Washington. *Fisheries* 16(2):4-21.

Nickelson, T. E., M. F. Solazzi, and S. L. Johnson. 1986. Use of hatchery coho salmon (*Oncorhynchus kisutch*) presmolts to rebuild wild populations in Oregon coastal streams. *Canadian Journal of Fisheries and Aquatic Sciences* 43:2443-2449.

NWPPC (Northwest Power Planning Council). 1992. *Columbia river basin fish and wildlife program: strategy for salmon*. Volume I. Northwest Power Planning Council. Portland, Oregon.

Page, G. W., P. L. Quinn, and J. C. Warriner. 1989. Comparison of the breeding of hand- and wild-reared Snowy Plovers. *Conservation Biology* 3:198-201.

Parsons, P. A. 1986. Evolutionary rates under environmental stress. *Evolutionary Biology* 21:311-347.

Powell, A. N. and F. J. Cuthbert. 1993. Augmenting small populations of plovers: an assessment of cross-fostering and captive-rearing. *Conservation Biology* 7:160-168.

Press, W. H., B. P. Flannery, S . A. Teukolsky, and W. T. Vetterling. 1989. *Numerical recipes in pascal: the art of scientific computing*. Cambridge University Press, Port Chester, New York.

Price, E. O. 1984. Behavioral aspects of animal domestication. *Quarterly Review of Biology* 59:1-32.

Ralls, K. and J. Ballou. 1986. Captive breeding programs for populations with a small number of founders. *Trends in Ecology and Evolution* 1:19-22.

RASP (Regional Assessment of Supplementation Project). 1992. *Supplementation in the Columbia Basin: Part II, supplementation theory*. Northwest Power Planning Council. Portland, Oregon.

Reisenbichler, R. R., and J. D. McIntyre. 1977. Genetic differences in growth and survival of juvenile hatchery and wild steelhead trout. *Journal of the Fisheries Research Board of Canada* 34:123-128.

Reisenbichler, R. R., and J. D. McIntyre. 1986. Requirements for integrating natural and artificial production of anadromous salmonids in the Pacific Northwest. Pages 365-374 in R. H. Stroud, editor. *Fish culture in fish management*. American Fisheries Society, Bethesda, Maryland.

Ryman, N. and L. Laikre. 1991. Effects of supportive breeding on the genetically effective population size. *Conservation Biology* 5: 325-329.

Salonius, K. and G. K. Iwama. 1993. Effects of early rearing environment on stress response, immune function, and disease resistance in juvenile coho (*Onchorhynchus kisutch*) and chinook salmon (*O. tshawytscha*). *Canadian Journal of Fisheries and Aquatic Sciences* 50:759-766.

Simons, T., S. K. Sherrod, M. W. Collopy, M. A. Jenkins. 1988. Restoring the Bald Eagle. *American Scientist* 76:253-260.

Spurway, H. 1952. Can wild animals be kept in captivity? *New Biology* 13:11-30.

Spurway, H. 1955. The causes of domestication: an attempt to integrate some ideas of Konrad Lorenz with evolution theory. *Journal of Genetics* 53:325-362.

Swain, D. P. and B. E. Riddell. 1990. Variation in agnostic behavior between newly emerged juveniles from hatchery and wild populations of coho salmon, *Oncorhynchus kisutch*. *Canadian Journal of Fisheries and Aquatic Sciences* 47:566-571.

Taubes, G. 1992. A dubious battle to save the Kemp's ridley sea turtle. *Science* 256:614-616.

USFWS (U. S. Fish and Wildlife Service), Western Region. 1991. *Pacific salmon management.* Portland, Oregon.

Verspoor, E. 1988. Reduced genetic variability in first generation hatchery populations of Atlantic salmon (*Salmo salar*). *Canadian Journal of Fisheries and Aquatic Sciences* 45:1686-1690.

Waples, R. S. 1991. Genetic interactions between hatchery and wild salmonids: Lessons from the Pacific Northwest. *Canadian Journal of Fisheries and Aquatic Sciences* 48(Supplement 1):124-133.